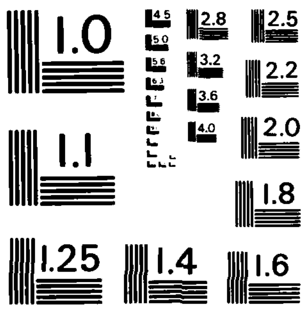
AD-A127 891 AN ENVIRONMENT FOR DISTRIBUTED SIMULATION OF COMMAND AND CONTROL NETWORKS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R A GRAHAM MAR 83 1/2

UNCLASSIFIED F/G 14/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN ENVIRONMENT FOR DISTRIBUTED SIMULATION OF COMMAND AND CONTROL NETWORKS

by

Richard A. Graham

March, 1983

Thesis Advisor: Bradford Mercer

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A127891	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Environment for Distributed Simulation of Command and Control Networks		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard A. Graham		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March, 1983
		13. NUMBER OF PAGES 106
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

command and control networks, distributed simulation, network simulation, command and control simulation, command and control testbeds, distributed testbeds

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis develops a concept for the simulation of command and control networks. The concept is based upon a model of the essential functions of command and control systems and networks of systems. The model is used as the basis for discussion of network performance evaluation, and the performance characteristics of concern form a basis for the simulation architecture. The simulation concept is based upon

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(20) a distributed simulation capable of utilizing a wide range of network node simulations ranging from manual procedures to manned simulators to fully automated emulators The simulation is both flexible and transportable due to it's residence within a computer based distributed environment

[illegible]	
[illegible]	[✓]
[illegible]	☐
[illegible]	☐
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or Special	
Dist	A

DD Form 1473
1 Jan 73
S/N 0102-014-6601

An Environment for Distributed Simulation of Command and Control Networks

by

Richard A. Graham
Captain, United States Marine Corps
B.S., The Ohio State University, 1975

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(Command, Control, and Communications)
March, 1983

MASTER OF SCIENCE IN COMPUTER SCIENCE
June, 1983

from the

NAVAL POSTGRADUATE SCHOOL

Author: [illegible]

Approved by: [illegible] Thesis Advisor

[illegible] Second Reader

[illegible]
Chairman, Command, Control and Communications Academic Group

[illegible]
Chairman, Department of Computer Science

[illegible]
Dean of Information and Policy Sciences

[illegible]
Academic Dean

ABSTRACT

This thesis develops a concept for the simulation of command and control networks. The concept is based upon a model of the essential functions of command and control systems and networks of systems. The model is used as the basis for discussion of network performance evaluation, and the performance characteristics of concern form a basis for the simulation architecture. The simulation concept is based upon a distributed simulation capable of utilizing a wide range of network node simulations ranging from manual procedures to manned simulators to fully automated emulators. The simulation is both flexible and transportable due to it's residence within a computer based distributed environment.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Several joint programs have been created in the past in order to answer questions regarding the interactions among the systems that comprise a command and control network. These include programs such as Tactical Air Control Systems/Tactical Air Defense Systems (TACS/TADS), Joint Interoperability of Tactical Command and Control Systems (JINTACCS), and Identification Friend, Foe, or Neutral (IFFN). These programs have been characterized by several similar types of shortcomings in their examination of the systems involved.

The first set of problems are those concerned with the commonality of the test systems used to conduct the network testing. A common set of air control and air defense systems are examined by each of the above programs. In fact, the JINTACCS program assumed the mission of the TACS/TADS program. Despite these obvious areas of overlapping interest, each of the programs required an entirely separate, but functionally similar, test system. As a result of the inability to share the necessary test capability, millions of dollars of development effort was repeated for each program. More importantly, the resolution of the underlying questions and operational problems was delayed for several years while the test system was developed.

A second set of problems are those concerning the staffing of these programs. Each of these programs has only a small staff. The full time job of this staff is to plan, conduct, and analyze tests of the target systems. In order to accomplish these functions, the staff must make decisions regarding a test scenario, experimental design, and data analysis. In designing the scenario, proficiency must be maintained in the operational doctrine and tactics of each of the systems under test, network procedures, limitations and distortions introduced by the simulations used for the test, and especially, the threat that can realistically be expected from the opposing forces. Personal observation would indicate that even the most highly qualified individuals tend to lose their proficiency in many of these areas after being assigned to one of the programs for a period of time.

B. SCOPE

1. Purpose

The purpose of this thesis is to present a structure for simulating networks of command and control systems which alleviates the above mentioned problems. Although the approach that is presented may be applicable to many similar problems at multiple levels of detail, it has been developed for a specific case. The development and the presentation here is focused on the examination of networks of tactical command and control systems. Since the emphasis is on the interactions among the systems which comprise networks, there was no attempt to incorporate a capability to examine the internal mechanics of individual systems.

2. Level of Presentation

The presentation of the methodology will be given at the level of a conceptual operating system task. A specific implementation for a given suite of equipment will not be provided. The presentation will give descriptions of functional characteristics, rather than specific methods for implementing these functions.* In many cases, the plausible or best implementation will be highly dependent upon the equipment selected.

C. ORGANIZATION

Sections II and III will present a model of command and control networks. The model that is developed will be directed toward identifying and understanding those elements of a command and control system which influence the characteristics of networks. In Section IV, this model will be used as the basis for discussing the evaluation of command and control network performance. Sections V through VIII will develop a simulation environment based upon the model of Sections II and III. The environment will be a computer based environment, but will be developed consistent with the idea of accommodating useful manual and semiautomated techniques into the simulation. The procedures and policies which must accompany the methodology will be presented in Section IX. And finally, Section X will summarize the results of the previous sections.

II. MODEL OF A COMMAND AND CONTROL SYSTEM

A. FUNCTIONS OF A COMMAND AND CONTROL SYSTEM

1. What is Command and Control

With the recent focus on Command and Control, there would hopefully be a universally acceptable definition or concept of what command and control is. Unfortunately, there is not. Therefore, virtually every paper that addresses the area of command and control must define what the author perceives the subject to be. Rather than taking the normal approach of listing numerous conflicting definitions and attempting to find the common elements, let us look at the functions performed by a command and control system. In this manner, it is possible to gain a feel for a command and control system without being unduly bound by a rigorous definition.

2. Determine the Environment

The first function performed by a command and control system is a determination of the state of its environment. This may be accomplished in a number of ways. The system may employ a radar unit to observe air and surface targets, or sonar to track underwater targets. It may receive digital data from other systems. In many cases this perception of the system's environment can be very simple, such as an individual scanning an area with his eyes. This last example highlights a very important concept. Note that although the equipment that comprises many command and control systems usually receives the most emphasis in descriptions and analyses, a command and control system does not have to be made up of computers and sophisticated sensors. It could simply be a platoon commander with his compass, map, notebook, and radio.

3. Formulate a Decision

Having formulated an impression of its environment, the system must formulate a decision based upon this impression. The decision reached could be to ignore the environment until a change of interest occurs. The system could decide not to act upon the information which it has gathered, but to forward the data to another system for possible action, or to hold it for future reference. And thirdly, the system may decide that some action, such as engaging a target, may be required in response to the environment. Of course, combinations of these decisions, such as taking action and forwarding the information are also possible.

4. Communicate the Decision

Once a decision other than to ignore the information has been made, it must be acted on. This requires the system to be able to communicate with other systems and/or to communicate with fire and maneuver units. Note that in an air defense missile battery, the missiles themselves are part of a weapons system and would not be part of the command and control system. However, this fire unit is very closely linked to the output of the command and control system's decision. Neither the command and control system, nor the fire unit would be effective without the other

B. ELEMENTS OF A COMMAND AND CONTROL SYSTEM

In order to accomplish the functions of determining the environment, formulating a decision, and communicating the decision, a command and control system can be considered as being composed of five elements: a set of sensors, a decision algorithm, a local data base, a method of communication, and some form of feedback. These elements are shown in Figure 1

1. <u>Sensors</u>

The term sensors brings to mind images of radars and sophisticated intelligence equipment. But sensors include all methods of observing the system's environment, from individuals to electronics. In addition, it is useful to consider radio receivers as sensors. This points out that sensors are used to form an impression of all aspects of the environment. Friendly forces and neutral aspects such as weather are as important to the perception of the environment as are the aspects associated with enemy forces. The receipt of reports and data from other systems

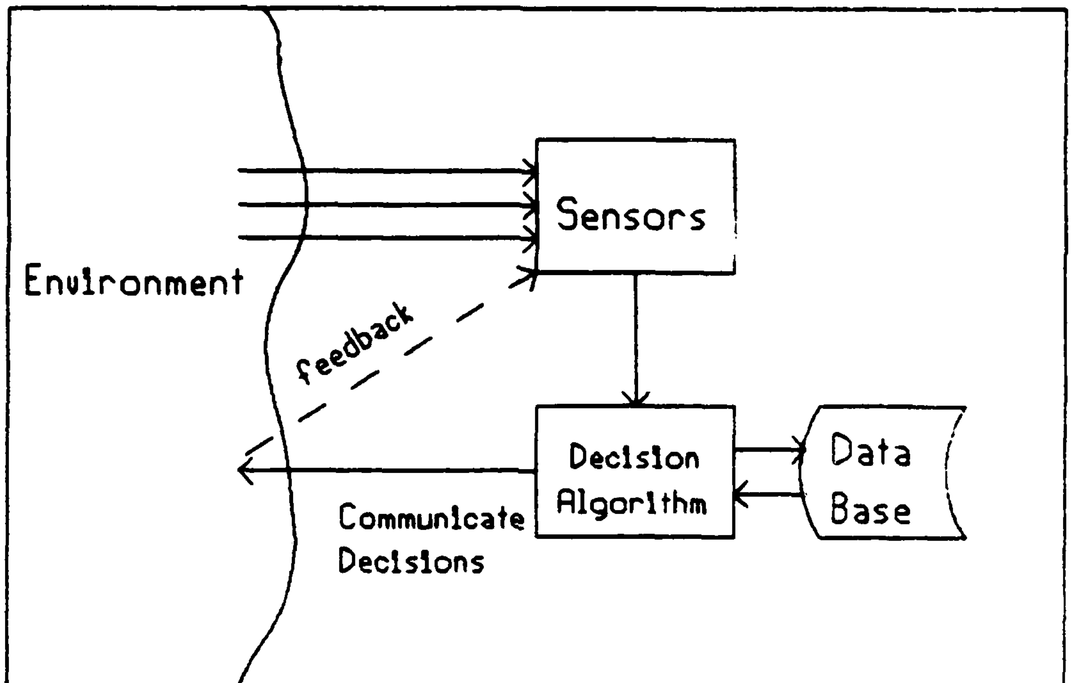


FIGURE 1. Elements of a Command and Control System

are also sensor inputs to the system. Sensors may be static in operation, or they may be dynamically controlled by the decision algorithm. An air defense system can typically employ both acquisition and fire control radars. The acquisition radar acts as a static sensor, reporting on all tracks within a fixed coverage. However, the fire control radar is used to track specific targets for engagement. The targets to be tracked are designated by the fire control system and thus the sensor is dynamically controlled.

2. **Decision Algorithm**

The decision algorithm used by a system is generally the most difficult portion of a system to define. It is a complex function of people, training, morale, procedures, doctrine, and equipment performance. Even when the decision algorithm appears to be solely the judgment of a single commander, it is usually influenced significantly by the manner in which the sensor data is manipulated after receipt and prior to presentation to the commander.

3. **Data Base**

Command and control systems employ a data base in order to store information deferred for future reference and to aid in the decision algorithm. This data base may be composed of an automated data base system, maps, overlays, or plotting boards. In systems involving personnel, it also includes the experience and knowledge of the individuals.

4. **Method of Communication**

Communication of decisions out of the system includes electronic transmitters, but it also includes verbal commands and reports. When "command by exception" procedures are included, such as those adopted for fire request coordination, a failure to communicate can also constitute communication of a decision.

5. Feedback

In order to be an effective control system, a command and control system must include feedback to determine the impact on the environment as a result of the system actions. Note that this feedback can be considered as a special component of the sensor inputs. Examples of feedback include explicit feedback, such as messages from maneuver units, and implicit feedback, such as radar observation of a change in an aircraft's speed or heading.

C. COMMAND AND CONTROL SYSTEM ENVIRONMENT

The discussion so far has centered on what functions and elements comprise a command and control system. As can be deduced from the preceding discussion, a command and control system is intimately tied to its environment. Therefore, a discussion of the components of the environment is now necessary.

1. Communications

As has been stated, communications receivers should be considered as sensors. Let us now consider the environment as a system using the model which has been used to describe the command and control system. Along this line of reasoning, the communication information received by the command and control system can be considered as an output of the environment system. By the same argument, the output of the command and control system decision can be regarded as a sensor input to the environment system.

2. <u>Perception versus Truth</u>

Continuing to consider the environment as a system, the data base can be considered to be the parameters of the real environment. However, the environmental parameters sensed by the command and control system are seldom totally accurate, and at best comprise only a small subset of the total parameters. Therefore, the decision process in the environment system can be considered to act upon the environmental parameters and produce an output to the command and control system.

3. <u>Environmental Dynamics</u>

The environment is not a static entity. It has dynamic characteristics that are due to the interaction with the command and control system and ones that are due to influences external to the command and control system.

a. **External Influences**

Dynamics due to external influences are changes such as the movement of units that are not controlled by the command and control system and communications inputs to the system.

b. **Response to Command and Control System**

The movement of units in response to commands from the command and control system and the resolution of engagements initiated by the command and control system are examples of dynamics due to the command and control system.

Both types of environmental dynamics can be thought of as direct results of a decision algorithm within the environment which responds to inputs from both the command and control system, and from other external sources.

D. GENERIC MODEL

The model presented so far is generally the same basic model of a command and control system that I have observed in most of the discussions and literature on command and control. The only difference lies in the consideration of the environment as a system itself. In this thesis, the emphasis is on the modeling, simulation, and evaluation of networks of systems, not the individual systems. In this context, a much simpler model of a single system can be used. This model is given in Figure 2. At first glance, this model appears to be too simplistic to be of any use. The decision algorithm and the data base have been merged into a single block, and the feedback element has become just one of several input/output paths, with no explicit correlation shown. However, as will be developed in subsequent sections, the only aspects of the single system which are of real importance in the context of networks of systems is the understanding that there is a finite set of inputs to the system from the environment, and that there is a finite set of outputs from the system to the

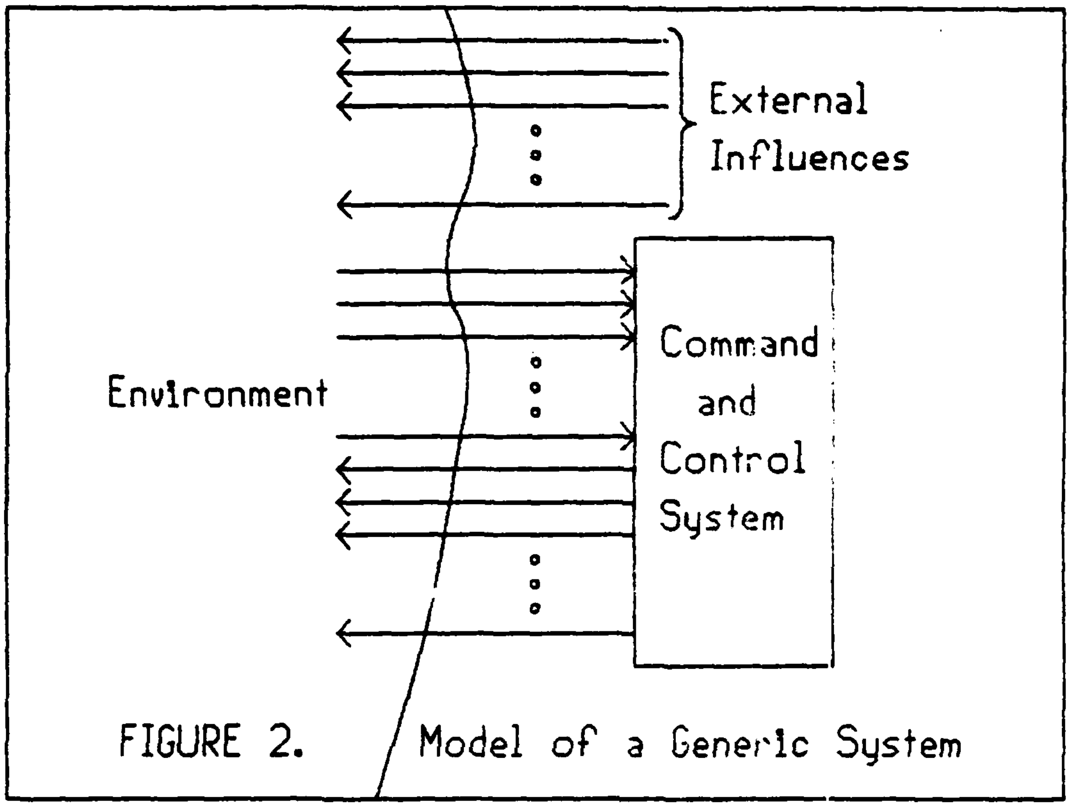


FIGURE 2. Model of a Generic System

environment. It is also important to recognize that while the system reacts only to the environment, the environment reacts to influences external to the system.

III. MODELING A NETWORK OF COMMAND AND CONTROL SYSTEMS

In this section, the model of an individual command and control system which was developed in Section II will be extended to address networks of systems.

A. A CLASSICAL MODEL

Consider a network of four systems, such as air defense systems, interconnected by both direct communications links and by a switched message system. Figure 3 shows how this arrangement would typically be portrayed. This type of representation is a digraph with decision elements represented as nodes, and communications paths portrayed by graph edges.

1. System Nodes

Three types of nodes are used in the graph. Each of the four systems is a single node. In addition, the communications switching system, and the external environment are represented by distinct nodes.

2. Communications

Communications paths are modeled as edges in the graph. However, the switched communications requires decisions to be made, and thus it is conveyed as both edges, representing the inputs and outputs, and as a node, representing the switching algorithm.

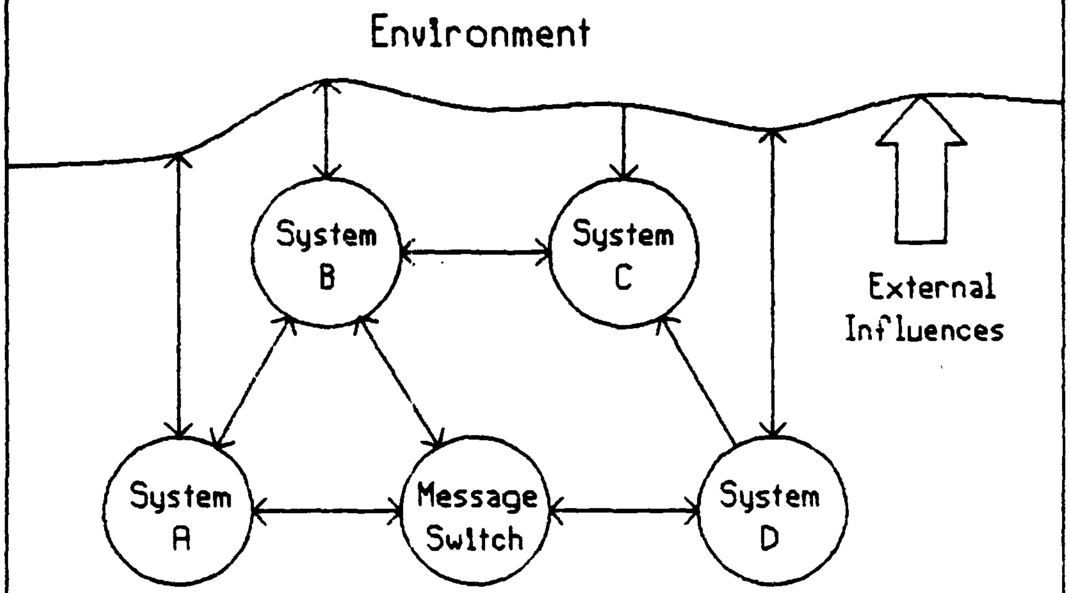


FIGURE 3. Classical Model of C^2 Network

3. __Environment__

The environment represents all of the items presented in the last section, except the communications mentioned above.

B. **AN ALTERNATE APPROACH**

This type of model suffers from two problems when used as a general model of a network of command and control systems. First, it is not really flexible. Each of the systems (nodes) has two types of edges, those that represent communications links, and those which represent interactions with the environment. When a system is added to or deleted from the network model, the model used for each node must be updated to reflect the changing edges representing the communications. Secondly, it is not compatible with the generic system model shown in Figure 2. Figure 4 presents an alternate model for representing this network arrangement.

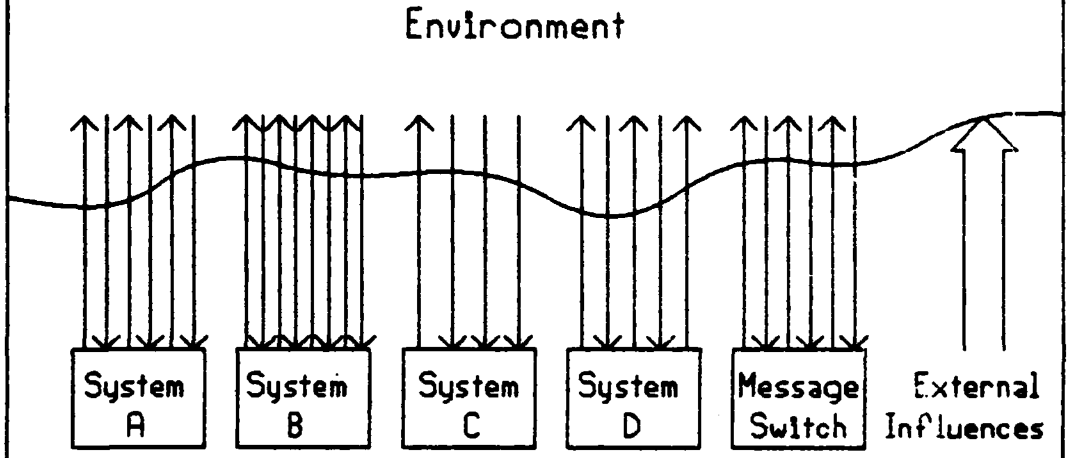


FIGURE 4. Modified Model of C^2 Network

1. __Linkages as Part of the Environment__

In this model, the linkages between systems are modeled as inputs from the environment to the systems and as outputs from the systems to the environment. Note that each interconnection is only in one direction and that each system interacts only with the environment. The model for each node can be viewed as in Figure 2. In addition, since there are no direct interconnections between systems, there is no need to change the model of the existing systems when a system is added to or deleted from the network, or the communications is rearranged.

2. Communications

In this model, it is necessary to enhance the model of the environment in order to convey all of the informaton found in the model of Figure 3. Specifically, the environment decision algorithm must include decision rules for the routing of communications data from an input to the environment to the appropriate output to a system. The best way to visualize the way in which the environment must be modeled is to examine the entire model from the perspective of the environment, as it views a single system.

C. THE EXTERNAL VIEW

1. Communications from the System

The communications coming from the system can be viewed simply as inputs to the environment, as in the single system model. The main point to be considered is that the system simply sends the data to the environment via one of its output paths. It is not necessary for the individual system model to know the destination of the data, other than a possible routing indicator for a switched communications system. Once the data reaches the environment model, it can be routed to the input of another system model, or it could be acted upon entirely within the environment model.

2. Communications to the System

The communications into the system are provided by the environment without the individual system model having to know the source, other than a possible indicator, such as a from data field, in the communicated information. Of course, the system also knows which sensor received the communication. The environment can obtain the data from either of two sources. It can be generated internally to the environment model, or it can be obtained as input from one of the systems. *Note that in the case where the data* is obtain from a system, it can be considered as an external influence to the environment from the single system perspective, as in Figure 2.

IV. COMMAND AND CONTROL NETWORK PERFORMANCE EVALUATION

This section will examine how the performance of a network of command and control systems can be evaluated.

A. A RESPONSE ORIENTED EVALUATION

In Section II, a command and control system was defined as a system that makes a decision based upon a perception of the environment, and then communicates this decision. The same definition can be applied to a network of systems. The only real difference is that a network of systems should hopefully be able to use the increased assets inherent in the network to develop a more accurate perception, thus yielding a "better" decision, and be able to communicate decisions more effectively.

1. A Stimulus and Response Model

Command and control systems and networks as they have been defined herein are an example of a stimulus and response system. The systems take their perception of the environment as a stimulus and respond with a decision. The stimulus and response can be traced by examining the data passing between the systems and the environment. The stimulus is the data input from the environment, and the response is reflected in the data passed to the environment by each system. The set comprised of all of the inputs to all of the systems constitutes the input to the network, and the set of outputs from all of the systems constitutes the response of the network. Generally, a single stimulus from the environment will result in a response from one system, which is provided, via the environment, as a stimulus to another system. The second system will then generate a response, and so on until the output of a system is sent to the environment and is not passed to another system. In this case, the network can be evaluated by examining the final response compared to the initial stimulus. In evaluating networks, two properties of the response should be considered; its correctness and its timeliness.

2 Correctness of Response

Based upon the evaluator's knowledge of the real environment, often referred to as "ground truth", a determination can be made as to the most appropriate response for the network to make. Note that even if the stimulus remains constant, different responses may be expected depending upon the purpose of the evaluation. If the objective is to develop tactics or doctrine, the response should be considered based upon its contribution to the achievement of a set of operational objectives. However, if the evaluation is made to determine the performance of a network within a given scenario or plan, the existing doctrine and tactics must be used as one of the measures of the correctness of the response, instead of as a variable.

3. Timeliness of Response

Each response has an associated time value. This response time is the time from the occurrence of the stimulus to the communication of the final response. The meaning of a specific time value must be weighed with respect to the value of the stimulus. A decision to assign a weapon system to engage an incoming missile must be made very rapidly. However, the detection of a surface ship at a range of several hundred miles may not require a time critical decision.

B. CONTRIBUTIONS TO CORRECTNESS OF RESPONSE

If the command and control network and the individual systems are to be evaluated based upon the response to a stimulus, it is necessary to identify the contributing factors in arriving at the response. This will allow for analysis of where corrective measures should be applied to the network in order to improve the response value or timeliness. The first area to be examined is the set of factors influencing the correctness of the response. As was stated previously, the response of the network is the result of the series of responses made by individual systems. Therefore, it is sufficient at this point to examine the factors contributing to the response from a single system.

1. Accuracy of Perception

The response to the stimulus is obviously based upon a perception of the nature of the stimulus, or environment in this case. Previously, it was stated that the system does not see the environment as it really exists. Instead, it only sees a subset of the characteristics, and may not see these characteristics accurately. In order to properly identify the stimulus, the system's perception of the environment must include all of the characteristics of the environment that influence the response. It is also equally important for these characteristics to be perceived accurately. If one of these two conditions is not met, it could be the result of either a deficiency in the system's sensors, or it could be the result of inaccurate data forwarded by another system. Note that one consequence of the latter case is that a system could provide accurate responses in one network, but fail totally in another network due to interactions with other systems. Thus, no system should be ignored in the analysis of a network simply because it performs well in other networks.

2. Availability of Supporting Information

Given that the system obtains a sufficiently accurate perception of the environment, it must use the local database to support the decision algorithm. The local database may or may not contain the needed data, and as with sensor data, it may or may not be accurate if it is present. Inaccurate information in the database can result from storing inaccurate sensor inputs or it can result from improper processing of previous inputs.

3. Decision Algorithm

The sensor and database data is acted upon by the decision algorithm in the system. Obviously, errors in this algorithm will introduce errors in the response.

C. CONTRIBUTIONS TO TIMELINESS OF RESPONSE

The timeliness of the response of a system is the sum of two factors, the time for the system to detect the occurrence of the stimulus, and the time required for the system to determine the response after the stimulus has been detected. An analysis of these factors usually includes a complex analysis of the communications delays in the network. But if the edges in the model are treated as instantaneous transfers of data, the communications delay analysis becomes a special case of the general system analysis, since the delays are introduced by either a communications system represented by a node in the model, or by the decision algorithm in the environment, which is modeled as a system.

1. <u>Routing through Network</u>

The time for the system to detect the stimulus can be generated by either or both of two factors. The first factor is the time required for other systems in the network to process and forward their results. The second factor is the time required for the system sensors to integrate the data once it is present at the sensor. For example, a normal search radar will average half of the rotation period in order to detect a target after it enters the coverage of the radar. Before, it was mentioned that a stimulus is in the form of a set of characteristics of the environment. The individual elements of this set do not always come from the same sensor or source, and thus could easily be subjected to different time delays in arriving at the system. Since all of the elements are necessary to the decision algorithm, the last of the elements to become available to the system determines the delay in detecting the stimulus.

2. <u>Time to Determine Response</u>

Within the system, the time required to arrive at a decision by the decision algorithm and to communicate this decision can be divided into three components.

a. **Input Queue**

The inputs from the sensors are queued in the system waiting for the decision algorithm to act upon them. This can easily be seen in a typical manual command post as the incoming message traffic piles up waiting for action officers to become available. Each system will have a different queue policy, ranging from a simple first in first out processing to complex prioritization schemes.

b. **Processing Time**

Each decision requires a finite amount of time for processing by the decision algorithm. Continuing the example of the command post, this represents the time during which the action officer is acting on the message.

c. **Output Queue**

Finally, the decision is subjected to another queue upon being released from the decision algorithm for communication. In the example, this is represented by the time required for the response message to be formatted and transmitted. It also includes the time spent waiting for other messages to be released ahead of the message of interest.

V. USING THE MODEL AS A BASIS FOR SIMULATION

This section shows how the previously presented model can be used as a basis for the simulation of a network of command and control systems. The discussion will center upon the use of computer technology to provide the basis of the simulation.

A. SINGLE SYSTEM SIMULATION

Recall the single system model of command and control that was presented in Figure 2. Using this model, a simulation of the system can be constructed with a logical structure as presented in Figure 5. It is important to note that this structure is designed to allow the analyst to view the individual system from a stimulus and response point of view. There is no desire at this level to examine the internal operation of the system. A complete analysis of the system would require additional tools to examine the internal dynamics of the system. The primary purpose of the structure in Figure 5 is as a building block to be used in simulating networks of systems.

There are three components in the simulation. The simulation of the environment will be discussed later, as will the interface process. The model and the simulation structure were developed with computer simulation in mind,

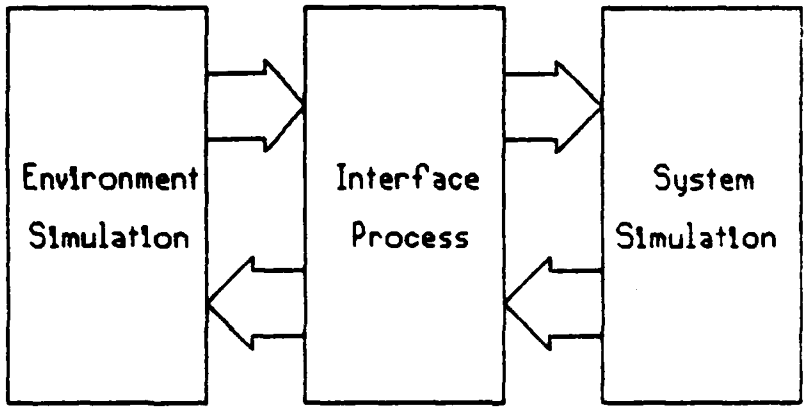


FIGURE 5. Single System Simulation

but it is not necessary for the simulation of the system to be a computer simulation. This is true even when the other two components are implemented on a computer. As presented in Section II, a single individual can be regarded as a command and control system. In this case, it would be much simpler, and probably much more realistic, to present information from the interface process directly to an individual for evaluation and to relay the responses back to the interface process. The only restriction that really applies to the system simulation is that it must be designed to accept the data provided by the interface process, and must provide output data in the format expected by the interface process. Other than this restriction, the simulation can be implemented by any available technique, to include mathematical models, computer simulation, manned simulators, or an actual system.

B. NETWORK SIMULATION

The extension of the single system simulation to the simulation of a network of systems will be presented by examining an example case. This example will then be extended to the general case. Consider a network of three command and control systems such as that shown in Figure 6.

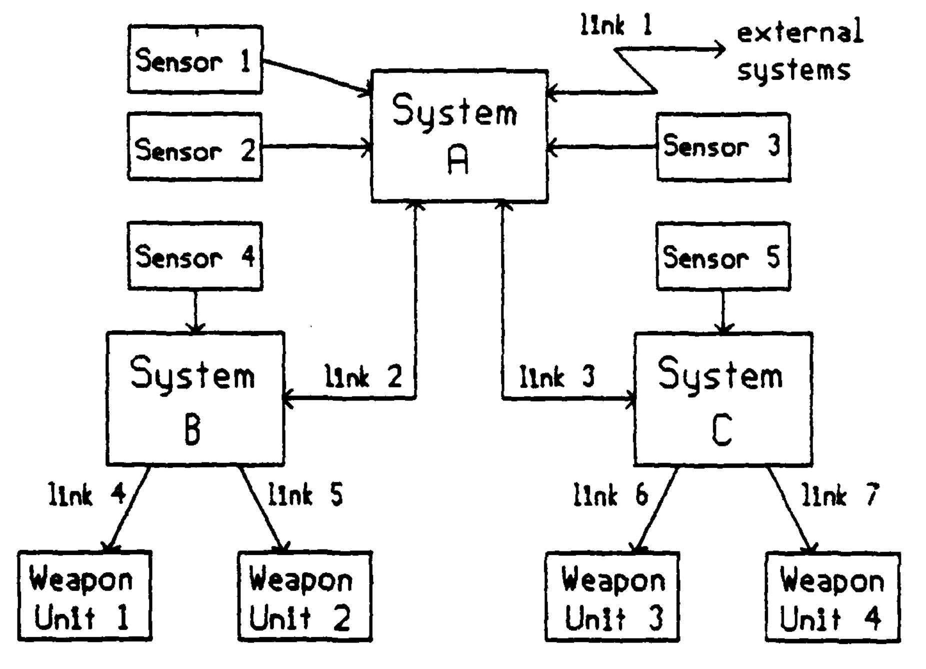


FIGURE 6. An Example Network

Using the model in Figure 4, and the general structure of Figure 5, we can simulate the network with the logical structure given in Figure 7. Each of the simulations for systems A, B, and C is as specified above. Note that the

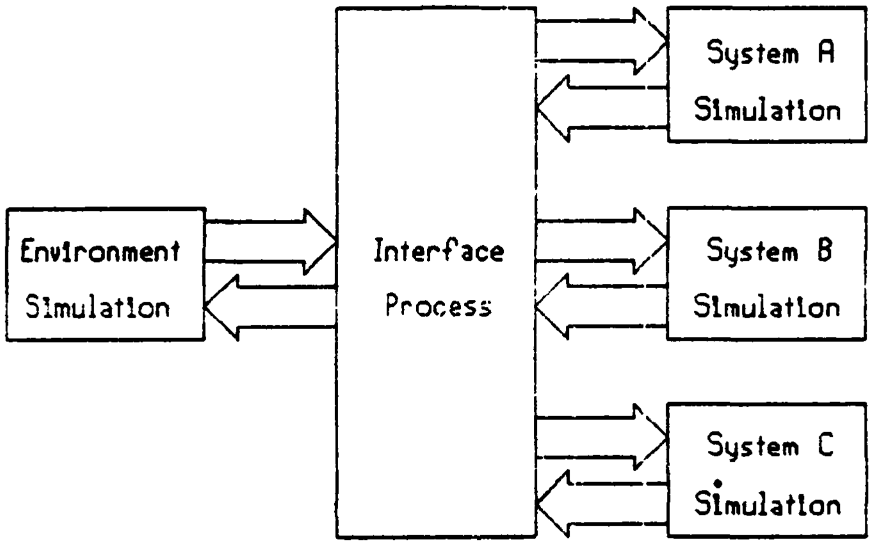


FIGURE 7. Simulation of Network

four weapons systems do not appear as part of the command and control network in the simulation. As discussed earlier, these components are part of the environment simulation. Thus, it becomes evident that the manner in which the environment is simulated will have a major impact on the validity of the entire simulation.

C. SIMULATION OF THE ENVIRONMENT

1. Items to be Simulated

What elements of the environment should be simulated in this example, and how should the simulation be structured? As stated above, each of the weapons systems must be simulated. It is also necessary to simulate the actions of opposing units and other friendly units. It may also be necessary to simulate such environmental factors as weather, time of day, and terrain if these factors influence the simulation of other elements such as unit movement or sensor performance.

One of the most difficult parts of the environment simulation will be in the handling of the system sensors for each system. As mentioned in previous sections, a realistic simulation of the various parameters of the environment is insufficient. Each sensor will have a view of only a subset of this environment. In addition, two sensors looking at the same subset of environmental parameters may still arrive at two different perceptions of the environment, since each sensor will not sense the parameters with the same accuracy. Different sensors will also assign different interpretations to the same parameter values. Thus it is necessary for the environment simulation to filter and modify the environment parameters before they are presented to the system simulations. This is especially true if the system is simulated by a person or part of an actual command and control system. In this case, the simulation of the system cannot be used to modify the parameters. If, on the other hand, a system simulation is designed strictly for use in the network simulation, the parameter filtering can be accomplished either as part of the environment simulation, or as a task within the system simulation. The former method provides more flexibility since many types of sensor technological changes can be accommodated with no change in the system simulation. An example of such a change could be an improvement in the accuracy of the location reported for a target.

Another component of the environment that must be considered is the set of characteristics of the seven communications links within the network. Since these characteristics may be variable, and have a major impact on the overall performance of the network, they must be accounted for. Depending upon the level of examination, it is reasonable to consider them as either part of the output task within a system simulation, or to represent the link as a separate simulation parameter in the environment. The latter appears to be a more general approach. It allows the channel characteristics to be modified to account for technological, environmental, or procedural change without having to modify the system simulation.

2. <u>**Simulation Methodology**</u>

Recall that earlier it was stated that the environment could be regarded as a system for purposes of modeling and analysis. When the proliferation of environmental parameters is examined, it appears to be more reasonable to consider the environment as a network of interacting systems. Since a method of simulating a network of systems has already been presented to simulate a command and control network, it seems reasonable to see if this same methodology could be applied to the simulation of the environment. When the methodology is extended, it results in a logical configuration like that in Figure 8.

Figure 8 reveals an interesting result of this approach. It no longer matters whether a given individual simulation or task is part of the environment, or whether it is part of the command and control network under examination. This is a rather subtle, but extremely powerful result. It means that if the structure presented can be implemented, then several apparently divergent problems can be solved economically as a group. Consider an analyst trying to determine the vulnerability of a U.S. command and control system to Soviet counter command and control measures. In this case, the U.S. system is the network under examination, and a network of Soviet systems is part of the environment. Now consider another analyst who is trying to find a vulnerability in the Soviet counter

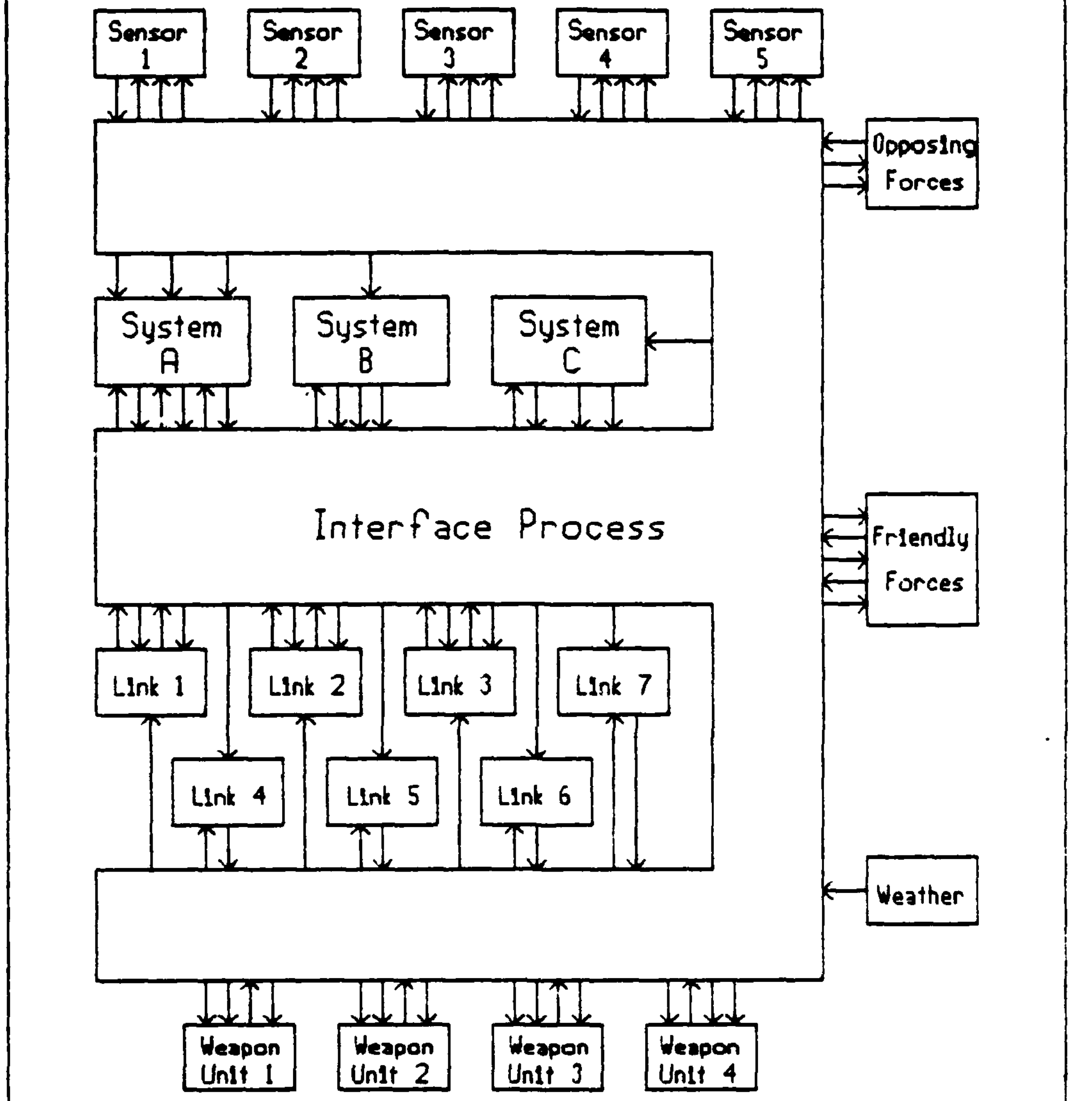


FIGURE 8. Tasks for Example Network

command and control measures. Now the Soviet network is the network under examination, and the U.S. systems are part of the environment. However, since the interface process doesn't need to distinguish between the network under examination and the tasks that comprise the environment, the same simulations could conceivably be used in both instances if they provide the proper level of simulation

D. THE INTERFACE PROCESS

It should have been apparent from Figure 8 that the interface process can very easily have numerous connections to a multitude of simulation tasks. In fact, every input to or output from a simulation task is connected to the interface process. As the hub of the entire simulation, the interface process is the prime determinant of the overall characteristics of the simulation environment. Environment in this case refers to the external structure within which the simulation exists, as opposed to the command and control environment that is simulated as part of the overall simulation. The majority of the remainder of this thesis will be devoted to the necessary characteristics and functions of this process

1. Functions

Figure 9 shows one of many possible configurations for the routing of data within the interface process for the example of Figure 8. Even for this relatively small example, the routing of information is obviously a major function to be implemented by the interface process. Note that there are two types of links. There are one-to-one links where the output of a single task is routed to

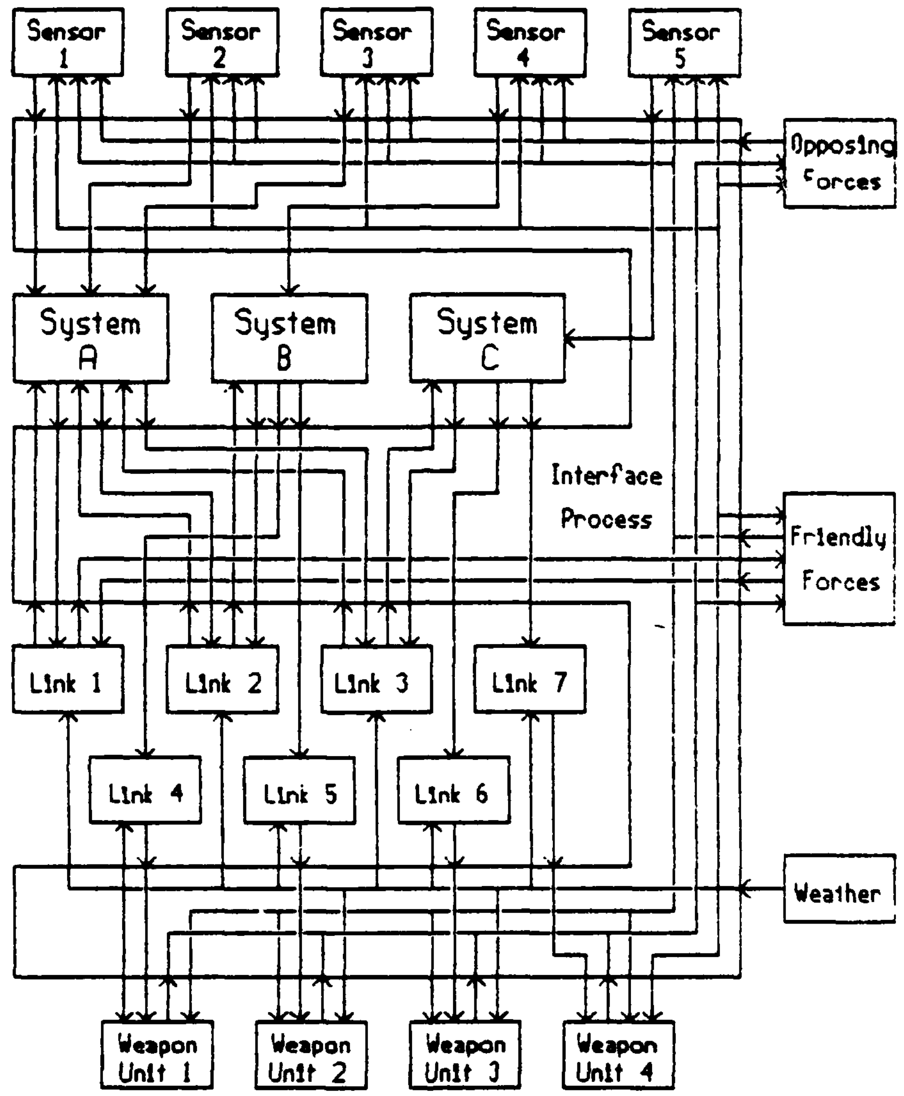


FIGURE 9. Example Links Between Tasks

another single task. There are also one-to-many links where the output of a single task goes to multiple destination tasks. There are also many-to-one and many-to-many links, but these are logically equivalent to sets of one-to-one or one-to-many links. Thus, the two types are sufficient.

Recall that the purpose of the simulation is to analyze a network of command and control systems with respect to a stimulus and response model as presented in Section IV. Recall also that all of the information necessary to accomplish this task is present at the inputs and outputs of the tasks comprising the simulation of the network. Thus, it is desirable that the interface process provide a way to capture the content of the logical linkages.

Obviously, it is not desirable to capture data passing across each logical link. Also, the linkages associated with a task are dependent upon the specific instance of its use in a simulation. It is necessary to establish and to modify the operation of the interface in a manner that is independent of the simulation tasks that are being interfaced. This means that the interface process must implement an interface control mechanism for use by the experiment controller.

2. Implementation

The following implementation is presented as a method for accomplishing the three fundamental interface process tasks of logical linkage, data capture, and control. At this point the assumption is made that the interface process is implemented on a computer system, as are each of the simulation tasks A methodology will be given in a later section for the case where the simulation tasks are not computer based.

Let the interface process be implemented as a process at the utility program or supervisor level in the computer system. This allows it to be accessible to all other tasks, and to have access to the computer system's I/O structure. The need for the I/O structure access will become obvious later in the thesis

Whenever a task begins execut.on, it accesses the interface process and passes a logical identification to the interface process. The method of accessing the interface process is obviously an item which is very dependent upon the computer system uti ized. It is crucial that the identification be unique. For instance, if an air defense battalion was being simulated, copies of the same task could be used for each of the firing batteries. However, each would require a separate logical identification so that the batteries could be distinguished by the control interface.

Whenever a task "opens" a communications link with the interface process, this link is assigned a logical name. For output links, the assigned name is the name of an address list with which the link should be associated. There is no requirement for a given address list to be unique to a particular output channel. Input links are named with a name that is unique within the task. This name is concatenated with the task identifier to obtain a unique channel name. One of the tasks to be accomplished by the control function will be the mapping of address lists to the channel names. To do so, the control function will maintain a list of one or more channel names to be used for each logical address. Whenever data is received from a task by the interface process, the data is sent to each input channel whose name is on the address list associated with the output channel. This allows both one-to-one and one-to-many links to be implemented by the same mechanism. It also allows data which is generated, but not needed in the specific simulation being considered to be discarded. This is accomplished by putting no entries in the address list. The interface process must allow a given channel name to appear in multiple address lists. This ensures that the logical many-to-one and many-to-many links can be implemented.

Returning to the example of Figure 9, consider the four weapon unit tasks. The tasks can be initiated with the logical names unit_1, unit_2, unit_3, and unit_4. If each of the units is simulated by the same program, the input channels will be assigned common logical link names. Let these names be weather, comm, and location for the inputs from the weather, link, and friendly forces tasks, respectively. The logical link names are concatenated with the task logical names to produce 12 unique channel names; unit_1.weather, unit_1.comm, unit_1.location, ..., and unit_4.location. The output from each weapon unit task can be assigned to address list damage. Since the names are not unique, the output from all four tasks is merged into a single data stream. If a separation had been desired, the tasks themselves would have to concatenate the task name, since only input names are automatically concatenated. The interface process knows that address list damage contains the channel names opposing.damage and friendly.damage. These are the input channels for the two forces tasks. The manner in which the interface process knows the members of the address list will be discussed later.

The address list scheme will also solve the data capture problem. All that is necessary to capture the data from a given link is to initiate a task that will operate on, or record, the data for a given link, and then to open an input channel from the interface process. Adding the channel name for the data capture task to the appropriate address lists will complete the function by capturing the data regardless of changes in the tasks associated with the link. Continuing with the example above, add a log task to the simulation with an input channel log.damage which is used to receive damage data for recording. Simply adding log.damage to the address list damage will allow the data to be copied at log in addition to the forces tasks.

The control function can be implemented in much the same manner. Design the control function as a collection of cooperating tasks. Figure 10 is a possible configuration for the interface process for some simulation

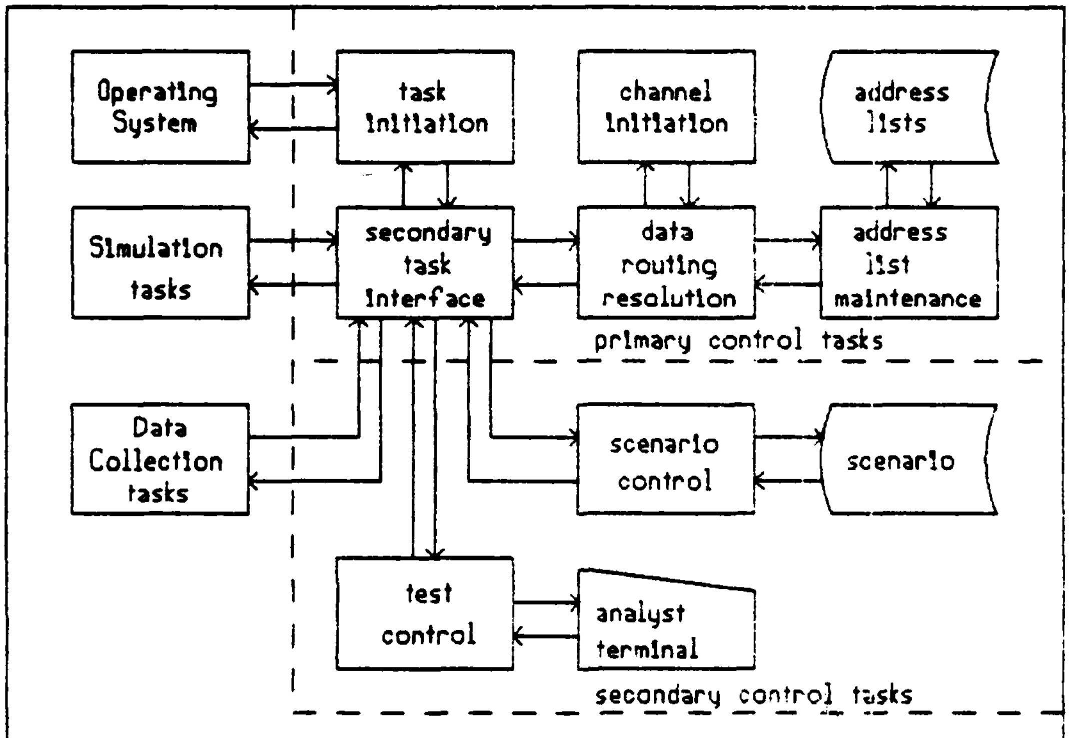


FIGURE 10. Interface Process Configuration

and host equipment. An individual task can be considered as being of one of two types. The first type is a task which must be present and logically connected to other tasks when the interface process begins execution. These tasks would include tasks which perform address list maintenance, tasks which physically route the information between input and output channels, tasks for initiating other tasks, and tasks for opening channels. These tasks will hereafter be called primary control tasks, or simply primary tasks. All other control tasks are secondary control tasks. These tasks include the tasks which interface the test controller to the interface process. (The test controller could be a human operator, a team of operators, or another computer process.) Assume that the linkages between primary tasks are fixed and are established at task creation time as in a conventional process. The linkages among the secondary control tasks and between the primary and secondary tasks can be implemented using the interface process itself. Let the input and output channels associated with each primary task be named by a fixed convention, and place each of the input channel names on at least one address list which is also named by convention. Each channel of each primary task is now accessible to any process that can access the interface process. This means that the exact configuration of the control function is determined by the selection of secondary control tasks, and their linkage to the primary tasks. Thus, it is easy to virtualize the control function to suit individual requirements and to add tasks as necessary to meet unique control problems for a given test environment.

3. Task Similarity

Note that as far as the interface process is concerned, the tasks that comprise the simulation, the tasks that implement the data capture function, and the secondary control functions all appear the same. This means that in subsequent discussion of task linkages, it is only necessary to discuss four cases;

a. primary task to primary task,

b. primary task to secondary task,

c. secondary task to primary task, and

d. secondary task to secondary task.

Since the latter three cases are implemented identically via the interface process, there are really only two cases to consider; primary task to primary task, and all others. Hereafter, the term secondary task will be used to collectively refer to all tasks other than primary tasks, regardless of functional purpose.

VI. A MODEL OF A DISTRIBUTED SIMULATION

In this section, the methodology of Section V will be extended to the distributed simulation of networks of command and control systems.

A. WHY DISTRIBUTED SIMULATION?

The simulation methodology presented in Section V is a logical network of tasks. This network could be implemented in a single computer, if the computer could support all of the processing required. An examination of the size and processing requirements of even a modest single system simulation leads to the inevitable conclusion that the representation of any significant level of detail in the systems comprising a network will result in an enormous processing load. The time needed to accomplish a simulation on a single computer makes a strong case for using multiple computers in a distributed processing arrangement. As used here, distributed processing refers to any collection of two or more processors processing components of the same problem. This includes colocated computers operating on a multiprocessor bus or local area network and also includes *geographically dispersed systems* connected by a long haul communications network. It also covers the hybrid systems resulting from networking geographically dispersed processing nodes. The individual processing nodes can be individual computers, local networks or a multiprocessor configuration.

Although the purely technical issues argue for a distributed arrangement, there are other strong reasons for a network. Recall the problems discussed briefly in Section 1. One of the problems was the amount of time that it takes for the development of a test system. In a distributed system, an organization with a functional requirement closely related to one already accommodated within the network only needs to develop minimal modifications, and a capability to access the network, in order to obtain an initial simulation capability. The technical aspects of the approach are presented in this section and in Sections VII and VIII. The management aspects are presented in Section IX.

The second problem area concerned the retention of technical expertise by the staff members of the organizations conducting the simulations. This problem is usually manifested in the area of scenario development. Due to the research and staffing that must go into a scenario in order to make it acceptable to all of the parties which are reviewing it, very few scenarios are ever developed by any organization. These are usually just variations of a single master scenario. The use of a distributed simulation with all of these organizations participating would allow the sharing of both the data bases that define the scenario environment, and the processes which control and distribute the scenarios.

A distributed network also provides a significant capability to organizations which normally would have no simulation capability. Analysis organizations within the services typically have insufficient computer assets for command and control network simulation. However, they usually have highly refined statistical analysis tools. By adding these organizations to a network, they gain access to simulations, and the organizations which are developing the simulations can concentrate on the simulations, knowing that the analysis tools have been debugged and have been made available for use.

B. LOCATION INDEPENDENCE

The key to making a distributed network feasible is to make the logical connectivity of a task independent of it's physical location. The solution is represented in Figure 11. The primary control tasks are augmented and are replicated in each of the network processors. Any task I/O that is passed to the interface process in any processor is passed to the proper tasks, in whichever processors they may reside.

Let the primary tasks in each processor have functional control over the communications between the processor and

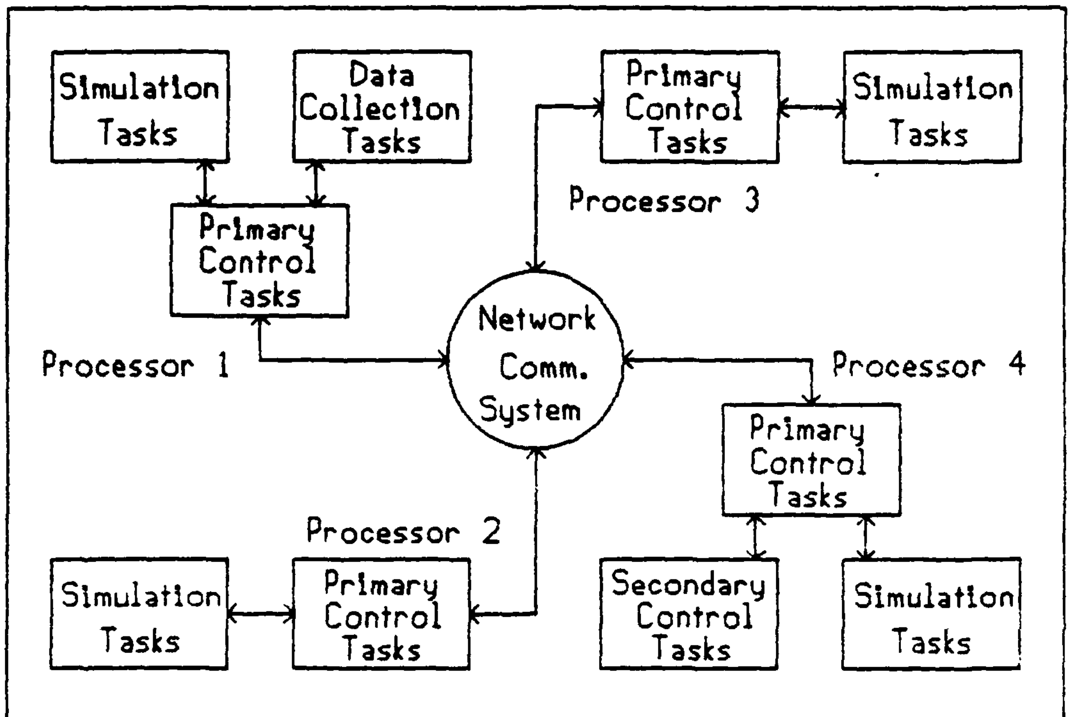


FIGURE 11. A Distributed Architecture

other processors. Note that it is not necessary for the primary tasks to directly control all of the links. The actual protocol and line handling could be implemented by an operating system utility or network communications package. It is essential though, that none of the secondary tasks has direct access to the communications links.

At least three methods are now possible for routing I/O to the proper task. The method actually utilized is a matter of a tradeoff between network complexity and flexibility.

The first method is the least complex, but also the least flexible. When a task is initiated, the logical address used to identify it within address lists will contain a processor identifier. Thus, the primary interface tasks always know the identity of the destination processors. Data is routed to each of the target processors, where it is passed to the primary interface tasks for routing to the proper tasks. Of course, efficiency of communications dictates that the transmission to multiple tasks at the same processor would only result in one data transmission, with multiple copies being generated at the receiving processor.

In the second methodology, every I/O transaction is broadcast to all other processors, addressed to the address list name. The primary tasks in each processor check the specified address list for local tasks and pass the data to tasks as appropriate. Many local networks and interprocessor buses use a broadcast method for distributing data on a ring or star network. When this is the case, this methodology is a very simple and flexible alternative. Of course, this method suffers from obvious deficiencies in geographically distributed networks with limited bandwidth between processors.

The third approach is for each processor to keep a copy of all address lists being used by local tasks. A special address list, named by convention, would broadcast to all processors as in method two above. One use of this special address list would be for the dissemination of changes to normal address lists. A secondary task, named by convention, would maintain a master copy of all address lists. When the primary tasks in a given processor needed to initiate a new address list, they would obtain the initial list from the secondary task. The lists would identify logical names and processors for each task. Although conceptually the most complex, this is the most flexible approach. It allows tasks to migrate among the processors from execution to execution of the simulation, or even during a given execution. At the same time, it minimizes the problem of channel bandwidth by keeping local subsets of the routing information. In the event of a failure that removes the master copy of all address lists from the network, the list can be reconstructed by merging all of the local subsets which remain in the surviving portion of the network.

Combinations of the above approaches are also feasible. For instance, either method one or method three could be used to route data between two local networks, and then method two could be used within each local network.

Note that only the primary tasks within the interface process need to be replicated within each processor. All other tasks are needed only once within the network, and may logically reside anywhere.

C. BEYOND A CLOSED SYSTEM

All of the tasks discussed so far have been assumed to have the ability to communicate with the interface process. However, there are simulation capabilities which must be considered for incorporation in any serious effort to simulate a command and control network, but which cannot communicate directly with the interface process. Two examples will be given.

Over the years, each of the services has developed a large inventory of manned simulators. These systems provide excellent capabilities for simulating both command and control systems and weapons systems. However, these systems generally reside on dedicated hardware, often without any operating system primitives beyond a simple monitor and bootstrap loader. It would be difficult, and probably imprudent, to modify them to incorporate the interface processes presented herein.

Fortunately most of these systems were designed for, or have been modified for, stimulation and monitoring by external computer systems. The term stimulation is often confused with the term simulation. A system is simulated by a process which models the actions produced in the simulated system. A system is stimulated by providing an input (or possibly a lack of an input) which causes the system to react in some manner, predictably or otherwise. These computer interfaces provide the key to utilizing these systems in the simulation network. A task can be created that maps the simulator interface to the interface required by the interface process, as shown in Figure 12. This task is executed in a computer colocated with the

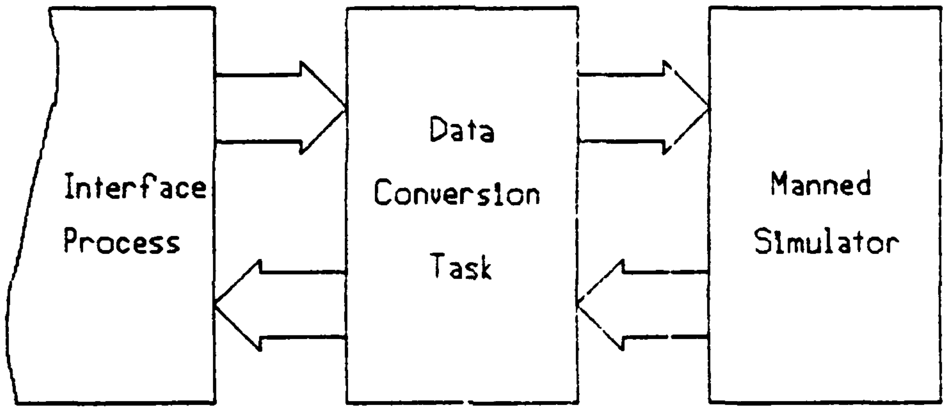


FIGURE 12. Utilizing Manned Simulators

simulator, and results in the simulator being addressed as if it was a single task, people and all.

The second example is when it is desired to monitor the exact exchange of data which is occurring between two tactical systems. In this case, the tactical interface must be as close to the tactical employment as possible. Use of the interface process would disturb the timing and handshaking characteristics of the interface and result in invalidation of the entire test. However, it is desired to collect the data and analyse it using network assets. The solution here is similar to that above, and is shown in Figure 13. The data is captured at the tactical interface

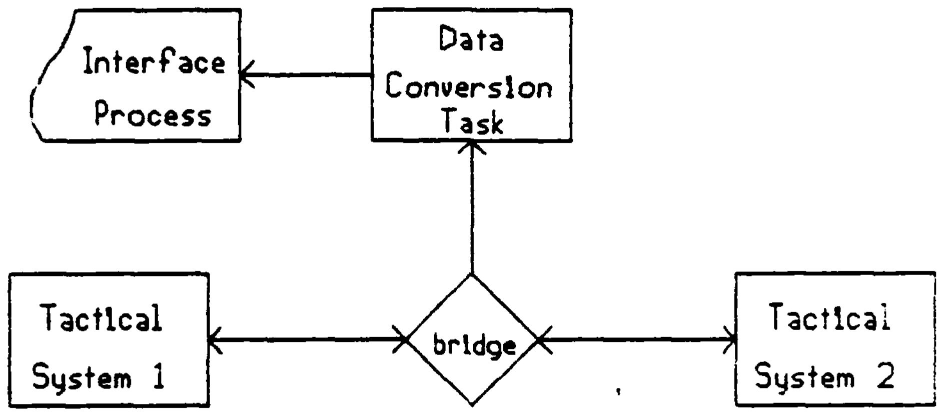


FIGURE 13. Capturing Tactical Data Streams

by any suitable means and is passed to a computer task that meets the interface process criteria. The data is now available throughout the network.

This basic methodology can be used for many tasks which are not directly suitable for the interface process, including simulations which are not computer based. Command posts are often simulated simply by a staff of people and some rudimentary communications for receiving stimuli and relaying decisions. If a computer terminal is used for the communications, possibly staffed by a test controller, then the stimuli can be released by a task in the network simulation, and the results feed back into the network. All that is needed is a suitable interface task connecting the interface process to the terminal at the command post simulation.

VII. CHARACTERISTICS OF SIMULATIONS

There are many existing simulations which could be adapted to the methodology presented in the previous sections. However, the basic characteristics of these simulations are dissimilar. These dissimilarities arise as a result of both the characteristics of the item being simulated, and as a result of the individual preferences of the simulation designer. These dissimilarities can be expected to continue in any tasks which might be developed specifically for the presented methodology. Two of the characteristics are significant to the simulation model. These are the time line basis of the simulation task, and the conventions that are used to pass information between the various tasks. These characteristics of simulations, and their impact on the simulation model, will be examined in this section.

A. TIME LINE CONSIDERATIONS

Simulations generally are time line classified into one of two categories, time step or event step.

1. Time Step Simulations

In a time step simulation, the task is given the state of the simulated phenomenon at a specific point in time. The task then calculates the state at a specific time interval later in time. As an example, consider the simulation of the movement of an aircraft. The task would know the aircraft's position, heading, rate of climb/descent, and velocity at some initial time, t_o. Based upon some time increment t_d, the task would compute the aircraft position at time t_o+t_d. During the next iteration, the position would be computed at time t_o+2t_d, and so forth. Typically, in this type of simulation, changes in heading, rate of climb/descent, or velocity can only be made at time t_o+it_d, for some nonnegative integer i.

2. <u>Event Step Simulation</u>

Note that if changes in heading, rate of climb/descent, or velocity occur infrequently, a period of linear aircraft motion is calculated as an accumulation of short segments of motion. If the time of the first aircraft maneuver is known to be $t_o+2000t_d$, the location of the aircraft when initiating this maneuver can be calculated directly from time t_o in one step by using $t'_d=2000t_d$. This requires considerably less work. This is the idea behind the event step simulation approach. A list of events (the maneuver in this example), and their time of occurrence is maintained. The state of the simulated phenomenon is calculated for the time of the next chronological event, based upon the state at the last event time. The prime disadvantage of this approach is that the time of occurrence of all events must be known in advance. This can often be very complicated to compute. For example, consider an early warning aircraft operating in an environment that includes several other aircraft and some opposing electronic warfare assets. As the various aircraft maneuver when will aircraft be detected or lost by the early warning radar?

B. INFORMATION PASSING CONVENTIONS

There is an obvious need for the various tasks within a simulation to pass information among themselves. The proper method for accomplishing this function is not so obvious. One of two methods is generally used. These methods are message passing and the use of a common data base.

1. Message Passing

One approach to data passing is to transfer items of data directly from one task to another. This can be implemented in a number of ways depending upon the specific equipment suite. One common method is the passing of parameters in a subroutine call. Another is the establishment of a queue where records are deposited by one task, and retrieved by another. The essence of the process is that task 1 calculates some element of data and sends it to task 2 directly across some logical channel of communication. This approach is simple, but there are some very rigid assumptions that must be met in order to avoid disaster. First, task 2 must be able to process the received data at a greater rate than it is produced by task 1. In some cases this may have to be accommodated by allowing task 2 to simply dump or ignore messages when overloaded. Secondly, task 1 must generate the data in the order in which it is needed by task 2. When the order in which task 2 requires data is not fixed, nor predictable by task 1, it becomes necessary to establish some sort of handshaking so that task 2 can pass it's requirements for data to task 1. This is usually implemented by a message passing channel from task 2 to task 1.

2. Common Data Bases

Consider the following tasks. Task 1 calculates the current position of a ship. Task 2 calculates helm commands for the same ship. It is obvious that task 1 requires the current location, heading, and speed of the ship in order to calculate a new location. Task 2 needs the location, heading, speed, and other data that possibly includes mission orders and the location of other vessels, in order to calculate a new heading or speed. In the common data base approach to information passing, a data base would be established that included at least the location, heading, and speed fields. Task 1 and task 2 would both have access to this data base. Whenever task 1 calculates a new location, it simply updates the location field of the data base. When task 2 needs a location for the ship, it simply reads the value in the location field. New helm commands result in a change to either the heading or the speed fields, and are immediately available to task 1. The two tasks can thus function asynchronously. The approach is not without problems however. Whenever task 2 needs a location value, it must query the data base because the value may have been updated. This doesn't cause a problem with two tasks and only 3 data fields. But as the number of tasks with common data fields increases and the size of the common data bases grows, it may be necessary to utilize secondary storage for these data bases. Depending upon the equipment and operating system used, these numerous references to the data bases can be severely limited by the bandwidth of the channels available for access to the data bases. Configuration management can also be a problem. Whenever the structure of a data base is modified, the location of data needed by a given task is modified There are several methods for minimizing this impact. The most popular approach in recent years is the use of tasks, often called information hiding modules, which accept logical data requests from the task and provide the logical to physical mapping and retrieve the desired data. Although simple in theory, this approach simply moves the point at which the modification must be made, and may have an impact on the bandwidth of the access channel.

C. HYBRID CONFIGURATIONS

As shown above, each of the time line and information passing methods has advantages and disadvantages, and none is best suited to all simulation tasks that are to be performed. Therefore, the network simulation methodology must accommodate tasks based upon each of these approaches if it is to be useful.

A important question is whether or not there is a legitimate need to mix both time line methods, or both information passing methods in a single simulation of a network. Consider the previous example of the early warning aircraft. It would appear that a time step simulation is best for the movement of the aircraft. Radar detection can be computed each time interval and the problem of predicting the time of detection is avoided. In many conceivable simulations, the movement of the hostile aircraft would be predetermined and the maneuvers stored in a scenario. The task which reads the scenario and sends the maneuvers to the time step aircraft motion task is most easily implemented as an event step task. Thus, the two time line methods might be encountered in a single network simulation.

Consider the simulation of two command and control systems which are tracking a number of aircraft. Further assume that the best method for the passing of location to the system simulations is via a common data base in this case. Now consider the simulation of a communication channel that carries track update messages between the two systems. An important aspect of this channel is the order in which the information is passed, since this affects the processing by the receiving system. Thus, the message passing methodology appears to be ideal for this case. In conclusion, both information passing methodologies might be present in a single network simulation.

D. MAKINC INTERFACES COMPATIBLE

In Section VI, it was stated that individual tasks could be developed independently, and that they could reside within different test system elements. These ideas will form the basis for examining the issues associated with the *interfacing of tasks*. The interfaces will be characterized into general types based upon the characteristics presented in this section.

All interfaces examined in this section will be assumed to be a one to one, single direction interface used to transfer information from task 1 to task 2, as shown in Figure 14. Note that the interface process is not shown, and it will simply be assumed for the remainder of this

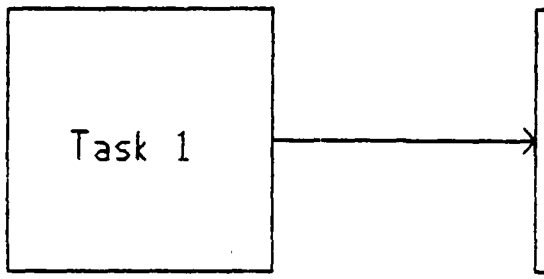


FIGURE 14. Task Interfacing

section. When two tasks are selected for use in a network simulation, they must satisfy a set of assumptions before an interface can be established. These assumptions are independent of whether or not the tasks are selected "off the shelf", or are developed for the simulation.

The set of inputs required by task 2 must be a subset of the outputs from task 1. Note that the sets do not have to be identical. When the output of task 1 contains, but exceeds, the input requirements for task 2, it is a simple matter to filter the output of task 1, as shown in Figure 15. When the output of task 1 does not contain all of the input required by task 2, the input deficiencies can be

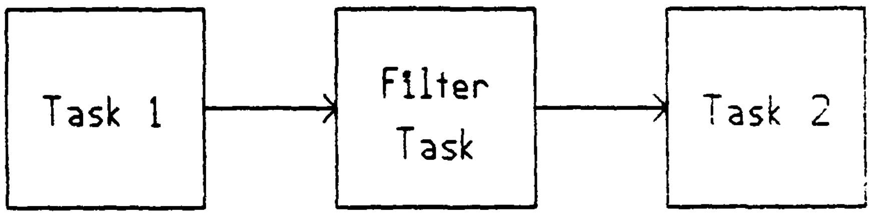


FIGURE 15. Task Interface Filtering

satisfied by one or more additional tasks as shown in Figure 16. However, for the remainder of the section, this case will be assumed to have been decomposed into two or more separate logical inputs to task 2.

The data input requirements of task 2 must be expressed in the same terms as the data output from task 1. As an example where this is not the case, task 1 might provide an aircraft altitude in meters, while task 2 expects to receive the altitude in feet. If the first assumption has been satisfied, this assumption can usually be satisfied by converting the improper data items to the desired terms, as shown in Figure 17.

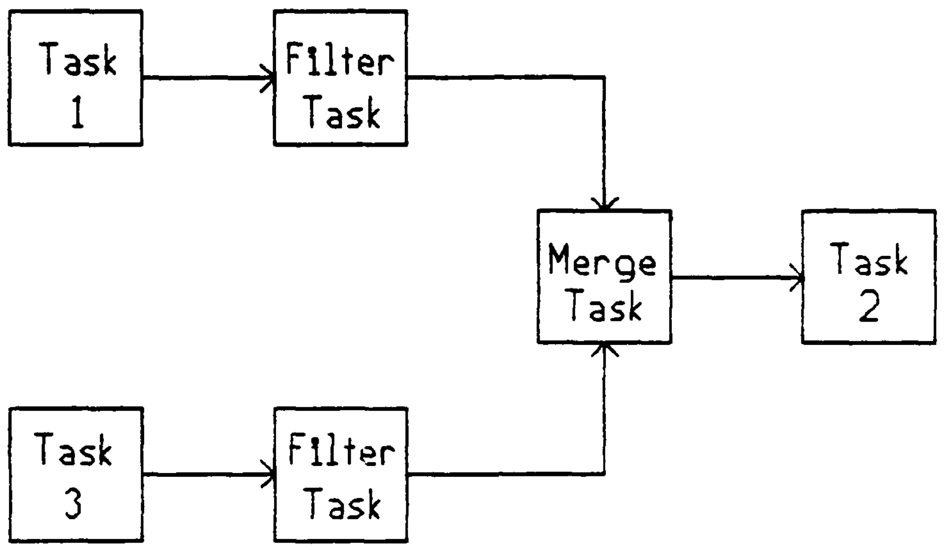


FIGURE 16. Composite Task Input

The remaining assumptions arise as a result of the characteristics of the individual tasks involved. The task characteristics will be examined separately for the time line and information passing combinations. This is possible because the assumptions required are independent and do not result from mutual considerations.

1. <u>**Time Line Interfacing**</u>

a. **Time Step to Time Step**

When two time step tasks are interfaced, the time step of task 2 should be at least as great as that of task 1. If not, the situation shown in Figure 18 can arise. The calculations performed by task 2 are

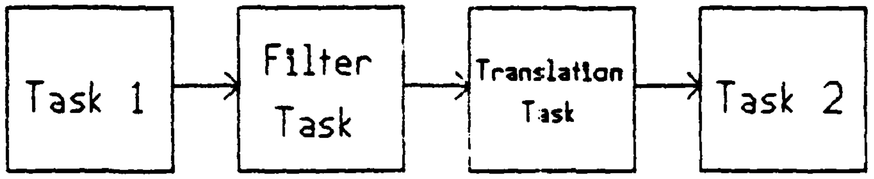


FIGURE 17. Task Interface Translation

accomplished using information which is not current. The ideal case is when both tasks have the same starting time, and the same time step. No time is wasted computing unneeded time steps by task 1, and task 2 always has the most current information. When task 2 has a greater time step than task 1, it should be much greater, or an integer multiple of, the task 1 time step. This will ensure current information in the case of an integer multiple, or information with insignificant aging in the case of much greater time step. When the step in task 2 exceeds the step in task 1, the filter task on the channel must be made aware of the relationship between the time steps. This is

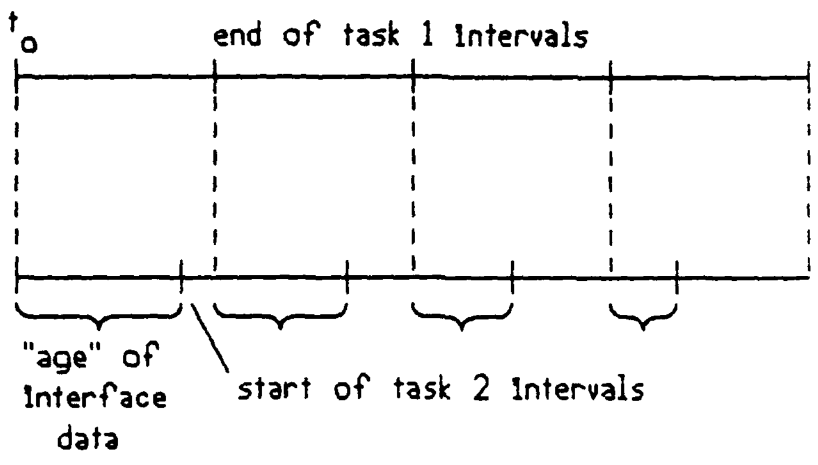


FIGURE 18. Currency of Time Step to Time Step Interface

necessary so that the unwanted time steps can be extracted from the task 1 output, while the current update is passed to task 2.

b. **Time Step to Event Step**

When a time step task passes data to an event step task, the translation task on the link must convert the output of task 1 into an event to be executed at a specified time. This can be accomplished in a couple of manners. The first is to treat each output from task 1 as an event to be executed at the end of the current time step. This essentially forces the event step task into the time step mode. This loses the efficiency that event step tasks gain from avoiding multiple identical calculations that sum to a result directly obtainable. The other approach is for the translation task to monitor successive outputs from task 1 and to only generate an event when there is a change in the data. In any case, this type of interface should be examined very carefully to avoid introducing side effects into the simulation.

c. Event Step to Event Step

There are no real concerns when interfacing event step tasks to event step tasks. This assumes, of course that the selection of the tasks is logically consistent with the simulation objective. The integration of several independently developed event step tasks is very difficult. This is because the essentially asynchronous manner in which the events are generated and executed makes side effects and logical inconsistencies very difficult to predict or resolve.

d. Event Step to Time Step

This case is like the time step to time step interface with a variable time step for task 1. As such, it is susceptible to the data currency problems when the step times are not matched. Since this mismatch is a transient condition, it can be frustrating to detect and correct. This type of interface is only recommended in one circumstance. This is when task 1 only affects data items which are discrete in time, and which cannot vary without task 1 producing a new output. As an example, the location of a ship is a parameter which varies continuously in time. If task 1 generates a location, and task 2 uses it, then there is a problem of task 2 not having current input values. However, the heading of the ship is usually treated as an instantaneously changing value, and is discrete in time. If task 1 generates a heading, task 2 can continue to use the same most recently received heading for several time steps without creating a problem. This assumes that task 1 will be executed whenever the heading changes. It may be necessary for the translation task to remember the most recent data from task 1 and repeat it for each time step of task 2.

e. Tasks Without Time Line

Some tasks have no time line. An example might be a task which given a missile type, a target ship class, and an impact aspect computes battle damage. These types of tasks are best thought of as subtasks within one or more tasks. Because they execute immediately and post results immediately, they generally present no problems other than the information passing problems discussed below.

2. Information Passing Interfacing

a. Message to Message

The interface process makes the interfacing of two tasks utilizing messages to pass information very easy. The interface between a task and the interface process is essentially accomplished by message passing. Thus, a message passing scheme is already present in each element of the simulation network. The only problems arise when task 1 expects to send messages directly to task 2, but task 2 expects to retrieve messages from a queue which buffers inputs. In this case, a task should be inserted into the logical channel in order to accept the messages from task 1, and to queue them for task 2.

b. **Message to Data Base**

The message passing task to common data base task is also very simple. Task 1 passes the information via message to the translation task, which must be coresident with the data base. The translation task inserts the data into the data base, where it is available to task 2, as given in section c following.

c. **Data Base to Data Base**

Problems arise in the data base to data base task interface. If task 1 and task 2 are not coresident, only one of the tasks, if either, can be coresident with the data base. The data base itself can be used as the filter task, but the translation task must still be accounted for. The solution is to create a data base access task for each of the tasks, as in Figure 19. Task 1 passes data base updates to access task 1, which is coresident with the data base, as messages. The access task performs any necessary translation and updates the

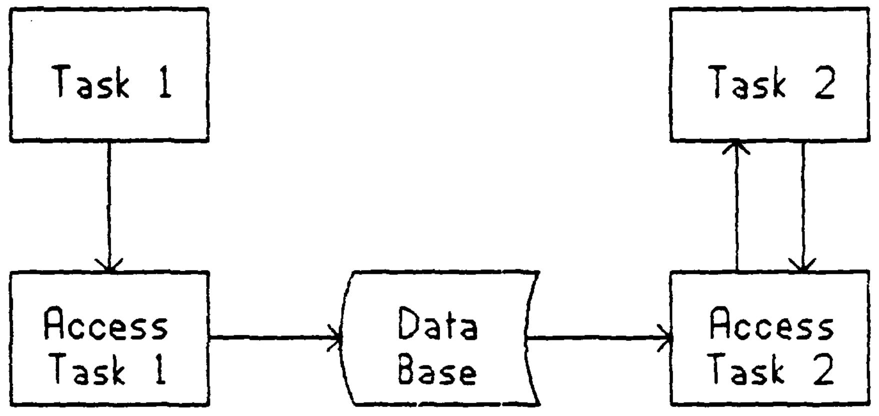


FIGURE 19. Data Base to Data Base Interface

data base. Task 2 requests desired data from access task 2 via a message. The access task reads the data base, translates the data, and sends the results via message to task 2. The portions of task 1 and task 2 which interface to the data base will have to be modified, but this is a problem that is inherent in using data base information passing. What is important is that the essential feature of the data base methodology, the data sequence independence between task 1 and task 2, has been preserved.

d. **Data Base to Message**

The most complicated interface to implement is when task 1 uses a common data base to pass information, and task 2 uses messages. Two cases exist. In the first task 2 expects certain information at specified times, regardless of whether the data has changed since the last message was received. This is typical of many tasks of a time step simulation. Figure 20 provides a general

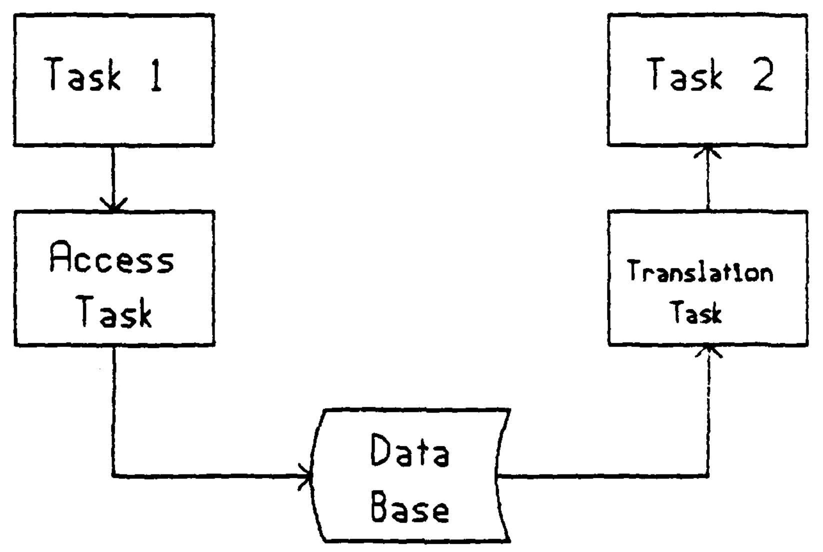


FIGURE 20. Data Base to Message Interface, case 1

approach. Task 1 updates the data base as described in section c. A translation task coresident with the data base reads the data base at the required time, and sends a message to task 2. In the second case, task 2 expects a message only when a data item has changed. In this case, shown in Figure 21, the translation task must be notified by the task 1 access task after a data item has been

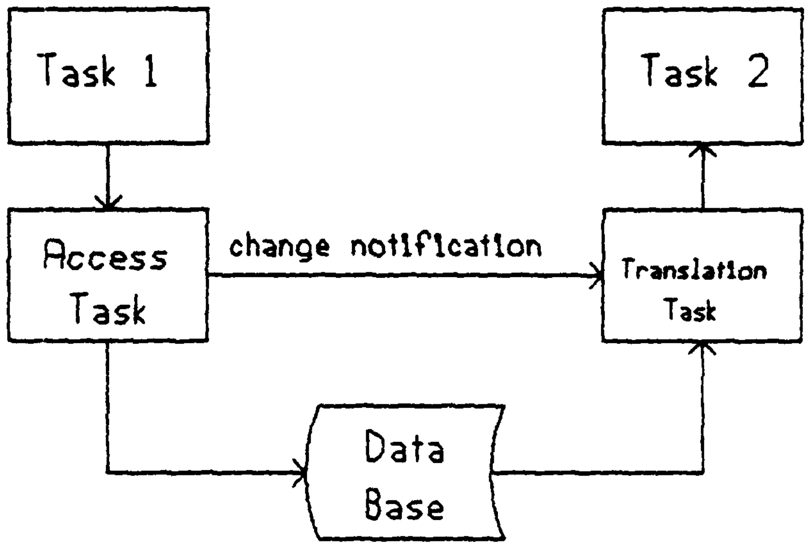


FIGURE 21. Data Base to Message Interface, case 2

modified. The translation task then collects the necessary data and sends a message to task 2.

VIII. REAL WORLD CONSIDERATIONS

There are several problems which will have to be addressed in each simulation implemented using a distributed simulation approach. The purpose of this section is to discuss a few of the more difficult of these common problem areas. It is important to realize that these are problems which must ultimately be addressed by the simulation designer, not by the network methodology. They are mentioned here primarily because they are common to many of the simulations which logically might be solved using the approach of a distributed simulation. Therefore, one or more of these problems will generally have to be solved before the approach can be used. These problems are also included for another reason. Experience would indicate that there is no solution to any of the problems which has general acceptance among the tactical command and control community. Thus, the manner in which these problems are addressed will be one of the prime determinants in establishing how useful a task will be to other community members.

A. SECURITY

Computer security in general is an incomplete discipline. There are several theoretical approaches ranging from security kernels to cryptographic systems. Unfortunately, there are not many applications of these theories to real systems. Those systems which are implemented are implemented as one of two types. In a single level system, every task has access to everything, because all of the information is at or below the classification to which the least cleared user has access. In existing multilevel systems, information is allowed to be migrated from lower levels of protection to higher levels, but the reverse is not true.

There have been attempts to avoid the security problems associated with command and control simulations by creating a test scenario with "test only" unclassified information. Unfortunately, there are several problems to this approach. Often the connectivity of the various nodes within a command and control network is classified. Hiding the connectivity by using test only network configurations could often invalidate the entire simulation. When actual tactical interfaces are involved, the information conveyed across the interface may be perishable and of low classification. Yet, almost all of the formats used for computer to computer interfaces, and many of the manual and computer assisted interfaces, are classified independent of classification of the content.

1. Downgrading and Sanitization

Accepting that a classified environment is unavoidable, is it possible to use an existing single or multilevel approach to solve the problem? The answer is no in some cases. The task of some of the nodes in many command and control networks is the declassification or sanitization of highly protected information for use by nodes with lower levels of access. A good example, without elaboration, is a direct support intelligence facility. This requires the capability in the simulation of the network for a task to make portions of information protected at some level available to tasks with an access level that would normally preclude access to the information. None of the current computer security systems accounts for this type of function.

The only approach with any current acceptability, that actually results in downgrading or sanitization, is the use of a manned simulator for the downgrading or sanitization task. In the distributed simulation, the manned simulator would be treated as a noncompatible task and would be connected to the interface process by two or more distinct tasks, one at each level of protection.

Even this approach does not provide a solution which is acceptable to all potential users. When the actual command and control system is employed in a tactical environment, there is a risk that a human operator or automated process will improperly downgrade or sanitize information and inadvertently compromise some portion of the original information. A major design and training concern in the real system is to maintain this risk at a level that is acceptable with regard to the consequences of not making the downgraded or sanitized information available. Many policymakers feel that a higher risk is acceptable in a tactical environment than is acceptable in a testing environment. Thus, any risk of operator or simulator error might be considered unacceptable. The only apparent solution to this assessment is the simulation of the system by two or more independent tasks. One type of task acts as a sink, accepting and acknowledging information at the higher level of protection. Another type of task generates representative information at the lower levels of protection, without any reference to the information received at higher levels of protection. The independence of the tasks means that there is no way to analyze any question which requires the flow of information through the simulated system, but a reasonable looking level of background activity can be generated for the loading of the network. This assists in the throughput analysis of other information transfer processes.

2. Classification by Collation

There is another security issue which must be addressed on a case by case basis. This issue is classification by collation. In these cases, a data base may be generated at a given classification level by collecting together a number of data items, all of which are individually classified at lower levels. For example, information regarding the location and characteristics of an individual target, such as a railroad bridge, may be unclassified. A target list containing information about all potential targets in an area of operations is a different matter. Each entry in the list could be unclassified, but their collation as a single entity provides information which exceeds the collective content of the individual entries. An examination of the list as a whole provides a picture of the overall accuracy of the targeting assessment for the area and, by exclusion from the list, of any oversights in the process. Therefore, the target list would be classified at a higher level. The resolution of this problem is a classic case of tradeoff analysis. If each entry in the list is protected at the level afforded to the list as a whole, it will not be available to the tasks which would normally have access to it. In addition, the task which initially generates the entry may not be authorized access to that level of information. On the other hand, treating the individual entries at their own classification introduces the risk of compromising the entire target list by allowing it to be collated one entry at a time at the artificially low level.

As stated before, each occurrence of classification by collation must be assessed individually.

3. Judgment

It should be obvious that there is no simple or clearly right answer to the security problems inherent in command and control systems and thus also in their simulation. Ultimately, the solution must rest with the judgment of competent authority, based upon an assessment of benefit versus acceptability of risk.

B. TIME SYNCHRONIZATION

Throughout the discussion of the distributed simulation there has been an underlying, but never stated, assumption that all of the tasks are essentially concurrent. There obviously must be some way to synchronize all of these tasks. A simulation task cannot begin to compute the result of a time or event step until all inputs from previous steps have been received from other tasks. There are several conceptual levels of synchronization, from multitasking to event step. Only two are particularly unique to the distributed simulation approach.

1. Master Clock

Numerous techniques can be utilized to synchronize the various tasks in a simulation. A large number use some sort of a master clock. This clock can be kept in terms of simulation time, event number, step number, or a number of different counters. The prime principle is that the clock is a globally available variable that synchronizes the task executions. This is a simple concept in theory, but the use of real computers in a distributed environment makes this a very difficult concept to implement. The individual tasks must be synchronized to the clock, a function which is addressed in section 2 below. In addition, the clock must be available to each task, which means it must be logically available at each processor. If there are ten processors, there are ten virtual views of the clock. synchronizing these views of the clock with each other will be addressed in section 3 below.

2. Synchronizing to Clock

Consider Figure 22. Task 1 computes parameters to be passed to task 2. Assume that both tasks are time step simulations, with the time steps synchronized to coincide. Task 1 and task 2 each signal task 3 when they have completed processing for the current time step. Task 3 then updates the clock. Task 2 detects the clock update either actively or passively and computes the next interval. This is simple in theory, mostly because all of the discussion, and the underlying model, have been based upon the assumption that the results of task 1 are instantaneously transferred to task 2. Unfortunately, the transfer takes a finite amount of time that is a complex function of the amount of information to be transferred,

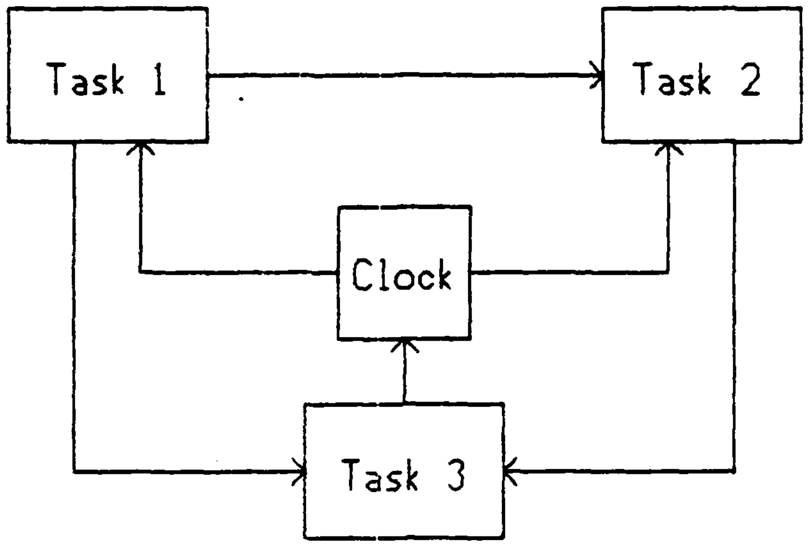


FIGURE 22. Task Synchronization

the data rate on the channel, and other processor and channel loading which must be multiplexed with the transfer function. The problem becomes even more complicated when all of the interface combinations discussed in Section VII are considered. Of utmost concern is the case where the transfer of information is via a data base to data base interface across three separate processors.

A lucky designer will be able to implement some sort of positive task handshaking, but shouldn't count on this solution in too many cases. Often, the clock is specified in the initial functional requirements to be tied to "wall clock" or real time. This is often necessary in order to connect to manned simulators or tactical systems themselves, or to accurately assess human performance under loading and fatigue conditions. The result is an exception to the general rule of location independence for tasks. Careful consideration and analysis must be applied to processor loading and to the required bandwidth between tasks. Tasks must then be allocated to processors in a manner which will allow transfer delays to be limited to times which are acceptably small compared to the step times expected.

3. Clock Synchronization

As stated above, communications is not an instantaneous process. Also, the delay between a given pair of tasks will vary with both time and the tasks comprising the pair. If two tasks in separate processors both query a clock variable in a data base in a third processor at the same time, they may receive different values. This is because one of the requests takes longer than the other to reach the appropriate data base access task. In addition, the responses are subjected to different delays and therefore, one of the times may be more accurate than the other when received by the original tasks. One example case will be used to show the complexity of the problem. Detailing all of the possibilities is not possible.

The worst case arises when the interprocessor delays are greater than the time step of the simulation. For example, consider the case where a tactical interface is being monitored utilizing the TADIL B interface standard. The characteristics of this standard are such that the monitoring of the data exchange must be time tagged to the nearest .001 second. The JINTACCS testbed, which tests this standard, stretches from Bedford, Massachusetts, to San Diego, California, using leased commercial telaphone lines. The communications delays in the circuits are much greater than the time step for the system clock. The master clock in this example would be some function of real time. The delay can be eliminated as a factor by keeping real time in each processor, along with the parameters that define the current translation function. However, this approach assumes that the real times kept by the processors are synchronized to within .001 second. Most computer system clocks are guaranteed to remain within some deviation from the initially set time for a specified duration of time. The problem thus reduces to one of accurately setting each clock at specified intervals.

WWV radio receivers could be used to receive the transmitted standard time signal. This approach suffers from two problems. First, it is costly to install and maintain receivers at a large number of sites which may participate only on an infrequent basis. Secondly, the receivers have oscillators and antennas. As such, unless expensive precautions are taken, they become an unacceptable security risk due to undesired electromagnetic radiation.

One solution is to arbitrarily designate one of the clocks as correct. The system is then configured so as to stabilize the delays as much as possible. Test messages are sent from the selected standard processor to each other processor and returned until a confidence interval is established for the round trip time. A message is then sent setting the remote clock to the current time plus half of the round trip delay. This message is also relayed back to the original processor in order to ensure against an abnormal delay during this message. This procedure works well if two assumptions can be satisfied. First, the confidence interval must be such that the maximum possible error in the one way delay time is less than the maximum acceptable clock error. The second assumption is that the time delay is the same in both directions. If not, a more complex set of test messages is necessary to establish delays in each direction. Once the local clock has been set, local tasks use this clock for synchronization and the delay from the master processor is no longer a factor until the clock must be reset.

C. SYSTEM VALIDATION

In any introductory statistics course, someone will always ask how big a sample must be in order to be big enough. The same is true of the testing that is necessary to validate a simulation once it has been designed and implemented. There is always considerable debate as to how much testing is enough. And, like the sampling question, there is no definitive answer. The testing of simulators and simulations is a very subjective and political issue, for a number of reasons.

Simulation validation is very expensive. The logical structure of a simulation can often be more complex than that of the system being simulated. This is due to the requirement to simulate complex phenomenon, such as weapons effects, which are taken as a given in the real systems. In addition, the simulation is interested in not only generating the same decisions as the simulated system, but also in tracing the evolution of the decision.

The result of validation is not a piece of hardware, or a program that can be utilized. Rather, it is some amount of confidence or doubt about the ability of the simulation to perform adequately. The cost factor and lack of tangible results lead to the subjective nature of validation. It is much like security in this manner. Ultimately, a commander must make an individual choice as to an acceptable and affordable level of validation.

The main concern with the distributed simulation is with the degree to which each task is validated. A major premise has been that tasks could be utilized off the shelf from other simulations. Using a task implies a certain degree of confidence in it's functioning. This must be based upon experience with the task, knowledge of the developing organization, or some validation process. Section IX will address configuration management within a network simulation community. It is essential that this configuration management effort track the validation associated with each task so that reasonable design decisions can be made regarding the task.

IX. MANAGEMENT CONSIDERATIONS

A sound management policy is essential to any complex computer system. In this section, several of the major considerations for the management of a simulation network will be presented.

A. CONFIGURATION CONTROL

Very few computer systems are static. The problems which the systems are designed to address often change. Perhaps more often, the user of the system may change his perception of the problem. Documentation and configuration management are the tools by which dynamic growth of a system are effected. Before a system is modified it is essential that the current state of the system be known. And as changes evolve, they must be tracked, so that unintentional side effects can be isolated at a later time.

In Section I, it was mentioned that the IFFN program was repeating development work done by the JINTACCS program, and that the JINTACCS program had repeated development work done by the TACS/TADS program. Why? Among other reasons, both the JINTACCS program, and the TACS/TADS program "saved" development funds by cutting back on the documentation and configuration management aspects of their respective test systems. One might assume that the importance of configuration management was not understood by the staffs of these programs. Yet both programs had a primary mission of managing the configuration management program for a Joint Chiefs of Staff directed interface among tactical command and control systems

The primary advantage to be gained from a simulation network is in the area of resource sharing. Before an off the shelf task can be incorporated into a developing simulation, the simulation designer must be able to understand what the task does, and what the interface specifications are. He must also be convinced that the task will remain available for use. This places a configuration management responsibility upon the organization which developed the task initially. The organization must provide complete documentation concerning the function of the task, the assumptions which have been applied, and the interface specifications. In addition, the organisation must baseline the task. A new "improved" version cannot be substituted for the task unless it continues to meet the needs of the other organizations which are using the baseline version.

A simulation network needs an active configuration management plan. Many approaches to configuration management have been shown to be successful. Any which is agreed upon by the network participants is acceptable, provided that it baselines each task, provides a method for approving new versions, and tracks the users of each task.

B. COMMUNITY ACCEPTABILITY

In order to be useful, a network would have to be accepted by the community which it is supposed to be serving. Past experience has shown that the Department of Defense is too fragmented, with too many loopholes, to force standardization compliance on any of the services, or their major elements. A look at past attempts to standardize programming languages, culminating in the recent decision to make ADA optional until it is acceptable to the user community, is a good example of how successful the "special case" argument can be. Therefore, a network will only be used seriously if it helps the using organizations accomplish their mission. The rest of this section is speculation, based upon experience, as to how acceptability could be achieved in a simulation network.

Initially, none of the major programs will share any of the assets available on a network. They have funds to develop unique systems, and a vested interest in producing results that are not reproducible by other organizations, particularly those which might be reviewing the results. However, a few of the well established and less paranoid organizations will make their tasks available on a network, if one is established. Some of these tasks, most probably those developed by the software support facilities within the services, will be documented in a usable manner. Analytical organizations within the services have generally been unable to afford large simulations, particularly of tactical systems. As a result, one or two will invest in access to the network. Using the assets available on the network, and providing some local tasks to customize the simulations to their particular needs, will provide an affordable new and powerful tool for use by these organizations.

These analytic organizations review and comment on the reports published by the major programs. As their comments become more detailed, and better supported, the tools which they are using will by examined by the major program managers. It is unfortunate, but true, that much more time is spent throwing stones at sound analysis methodologies than in discussion of the results of the analysis.

As a result of this examination of the capabilities of the analytic organisations, the major programs will begin to utilize the assets of the network in order to provide a capability to rapidly repeat and alter the tests conducted by the reviewing agencies. There seems to be an institutional rule at the joint service level that says that if a simulation can be shown to fail under one set of assumptions, it is invalid for any other set of assumptions. A very common rebuttal technique when reviewing the results of a simulation is to modify the assumptions, and show that the result with the new assumptions is absurd. Thus, the original analysis must also be absurd. This may not be a very valid argument, but experience has shown that it is very prevalent. Correct or otherwise, it provides a powerful momentum for having access to the same tools which are being used to generate the original results. As a result, when the organizations with tight budgets have accepted a network as a valid tool, the more affluent programs will accept the concept as a means of self defense.

C. TACIT STANDARDIZATION

As indicated above, efforts to enforce standardization on operational military systems have not been particularly successful. In this light, it would appear to be even less likely to enforce a standardization of the tools used to develop and analyze these systems. Yet, the entire concept of the simulation network rests upon the standardization of the interfaces among tasks. How can this paradox be resolved?

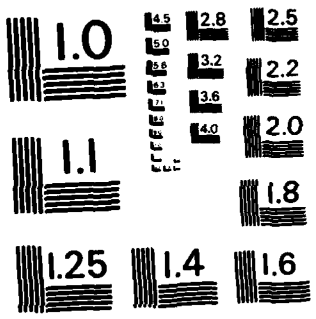
The previous section presented speculation as to how a network might come to be accepted by a community of users. If this premise is accepted, it follows that there will be one or more sets of tasks which comprise a core simulation with a family of tasks to customize this core to particular

AD-A127 891 AN ENVIRONMENT FOR DISTRIBUTED SIMULATION OF COMMAND AND CONTROL NETWORKS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R A GRAHAM MAR 83

2/2

UNCLASSIFIED F/G 14/2 NL

END
DATE FILMED
5 83
DTIC


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

applications. Since these tasks would be written to conform to the existing interfaces in the set of core tasks, these tasks comprise a tacit or de facto standard. The interface is controlled by the configuration management process and is "approved" by usage.

The prime difference between these standards and an enforced standard is that enforced standards require the foresight to develop the standard prior to development of the tasks to be supported. The de facto standard is an evolutionary result of the usefulness of a particular set of tasks. There are numerous examples of de facto standards in both industry and the military.

The CP/M operating system for microcomputers is not a particularly elegant, nor efficient, operating system. Yet, it was useful and available and became widespread. The large amount of supporting software that was developed using it established a de facto standard for commercial software written for 8080 family microcomputers.

The ARPANET provides an example within the military. There are numerous editors available on the network. Several of the more useful have been utilized as callable tasks by other processes, such as message handlers, and as front end processors for text formatting programs. These program linkages and dependencies form a de facto standardization on a distributed network. In this case, the community which has embraced this standard has a number of diverging mission areas. But, the standard is useful, and therefore it has survived.

D. SCHEDULING

One of the greatest problems in the simulation of command and control networks is scheduling. Several of the organizations which have the major assets, particularly the manned simulators, are working three shifts a day. The use of a network probably will not alleviate the work load on these simulators. The only savings to be gained would be the freeing of the simulator when another form of simulation, previously unavailable, could satisfy the requirements of a particular test. However, this would be offset by the requests from organizations which previously did not have access to the simulator.

There is one case in which the network could provide some relief on tight schedules. A given simulation task can often be conducted at any of several sites. This applies to both the manned simulators, which are often situated at both development and training sites, and to computer programs, which could run on any of a number of similar configurations. Typically however, they are only run in one location. This is because only one location has the analysts and test control personnel to supervise the test and dissect the results. Using a network, the logical availability and connectivity of a task are not dependent upon its physical location. Thus, it becomes possible to relocate a given task to a new site, with no analytical capability loss. There is an impact on test control only if a controller must be physically located with a test participant due to the nature of the simulation.

X. SUMMARY

This section will present a short summary of the important requirements for a distributed network to support command and control network simulations, and some personal comments regarding the methodology.

A. REVIEW OF REQUIREMENTS

Although the methodology presented in this thesis can have very powerful results, it is reasonably simple in concept. Therefore, there are very few hard requirements for implementation. The following items are the key requirements presented in the previous sections.

1. The processing nodes in a command and control network should be modeled as separate systems without any assumptions about the actual source of input information or destination of output information. This allows the level of detail in the simulation, and the configuration of the simulated network to be varied without impact on the individual node simulations.

2. All simulation tasks should communicate via an interface process. This process resides at the utility program or operating system level of the host architecture. As such, it has access to all of the I/O capabilities of the host machine.

3. The interface process is segmented into primary and secondary tasks. The primary tasks are essential to the function of the process, and are bound to each other at the time of creation The secondary tasks tailor the interface process to the individual requirements of a given simulation.

4. Only the primary tasks must be replicated in each processing node in order to support a distributed architecture.

5. Individual tasks send data to logical address lists, not to specific tasks. The address list associated with a task's channel of communication is established when the channel is opened and remains constant.

6. Entries in address lists may be dynamically changed during the execution of a simulation. Address lists contain the logical names of the task input channels designated to receive data sent to the address list.

7. The primary tasks within the interface process are responsible for translating logical channel names into communication paths with physical tasks. The primary tasks are the only tasks aware of the relationship between address lists and logical names, and between logical names and physical channels.

8. Several issues to be resolved by simulation developers, such as security, will be major factors in the usability of the network by a large community of users.

9. Documentation and configuration management are absolutely vital to the degree to which network resources are sharable.

10. Standardization should evolve as a function of task usefulness, rather than being imposed as an entry condition.

B. COMMENTS

The methodology for the implementation of the interface process which has been presented in this thesis has been addressed from the point of view of supporting the simulation of command and control networks. However, it is really a very general system concept. As such, it appears to be useful in a large number of networking environments where distributed tasks must act cooperatively. For instance, consider a distributed information management system. One of the major problems to be addressed has always been how to cope with a dynamic network configuration so that information reporting, collating, or retrieval processes know where data bases are. Several prototypes simply give each process a list of alternative locations to be searched in order to locate the data base. A much simpler approach might be to use logical names for the data base access tasks and allow an *interface process* to resolve the location problem. Updating a logical data base with new or changed data is also simplified, since the dynamic nature of the address list allows multiple updates from a process as easily as a single update. And, the data capture capabilities accommodate the requirement to save copies of transactions for recovery and reconstitution purposes.

Several people who have reviewed draft copies of portions of this thesis have asked questions such as "How would the tasks associated with the simulation of . . . be distributed across the network?" This thesis has attempted to specifically avoid addressing that type of question. This policy is not because of the difficulty of answering the question, although it is a difficult question. Rather it is a matter of the proper perspective on the purpose of the thesis. The purpose of the thesis was to present a simulation environment that would support a distributed implementation with the sharing of resources. Hopefully, it does so. However, while it is important that the specification address all aspects of importance, it is also essential that it not be overspecified. This environment is designed to help the simulation designer by extending the options available to him. An attempt to answer issues such as that above would only serve to place artificial limits on application techniques utilized.

For the same reason, I have failed to address the issues of how the primary tasks should communicate with the secondary tasks, and how the interprocessor communications should be managed. Neither issue impacts on the basic concept. Any method which makes sense for the particular equipment configuration utilised will support the concept.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Superintendent ATTN: Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Chairman Command, Control and Communications Academic Group Code 74 Naval Postgraduate School Monterey, California 93940	2
4.	Department Chairman Department of Computer Science Code 52 Naval Postgraduate School Monterey, California 93940	1
5.	Capt. Richard Graham, USMC SMC 2458 Naval Postgraduate School Monterey, California 93940	2
6.	Capt. Bradford Mercer, USAF Code 52Zi Naval Postgraduate School Monterey, California 93940	2
7.	Dr. J. Wozencraft Code 62Wz Naval Postgraduate School Monterey, California 93940	1
8.	CDR Gary Porter, USN Code 55Pt Naval Postgraduate School Monterey, California 93940	1

9. LtCol Joseph Mullane, USMC 1
Code 0309
Naval Postgraduate School
Monterey, California 93940

10. Director, Long Range Planning 1
Office of the Deputy Under Secretary
of Defense (C^3I)
Room 3E182, The Pentagon
Washington, D.C. 20301

11. Deputy Director, Defense, Test 1
and Evaluation
Office of the Under Secretary
of Defense (Research & Engineering)
Room 3D973, The Pentagon
Washington, D.C. 20301

12. Como. Rodger Simon 1
Code 06
Naval Electronics Systems Command
Washington, D.C. 20363

13. Dr. Joel S. Lawson, Jr 2
Code 06
Naval Electronics Systems Command
Washington, D.C. 20363

14. Capt. Dennis Glover, USN 1
Code 61
Naval Electronics Systems Command
Washington, D.C. 20363

15. Mr. Joseph Volpe 2
Code 62
Naval Electronics Systems Command
Washington, D.C. 20363

16. Institute for Defense Analysis 1
1801 North Beauregard Street
Alexandria, Virginia 22311

17. Capt. James Darwin, USN 1
JTF-IFFN
Kirtland AFB, New Mexico 87117

18. Dr. Edward W. Page 1
Computer Sciences Department
School of Nursing Building
Clemson University
Clemson, South Carolina 29631

19. Commanding General 1
Development Center
Attn: Mr. Druisback
Marine Corps Development and
Education Command
Quantico, Virginia 22134

20. Commanding Officer 1
Marine Corps Tactical Systems
Support Activity
Marine Corps Base
Camp Pendleton, California 92055

21. Commandant of the Marine Corps 2
Code CCA
Headquarters, United States Marine Corps
Washington, D.C. 20380

22. Commandant of the Marine Corps 1
Code CCI
Headquarters, United States Marine Corps
Washington, D.C. 20380

23. Commandant of the Marine Corps 1
Code CCT
Headquarters, United States Marine Corps
Washington, D.C. 20380

24. Mr. Dennis McCall 1
Code 8242
Navy Ocean Systems Center
271 Catalina Blvd.
San Diego, California 92152

25. Mr. Henry Sowizral 1
The Rand Corporation
1700 Main Street
Santa Monica, California 90406

LME

8